МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра информационных систем управления

ГУМИЛЕВСКИЙ
Кирилл Сергеевич

ПРОЦЕСС ТРЕНИРОВКИ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ МЕТОДОМ ОБУЧЕНИЯ БЕЗ УЧИТЕЛЯ ДЛЯ РЕШЕНИЯ ЗАДАЧ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Руководитель:
Прокопович Г.А.
доцент кафедры ИСУ

Рецензент:
Недзьведь А.М.
д.р. техн. наук

Минск
2020

АННОТАЦИЯ

Гумилевский К.С. Процесс тренировки искусственных нейронных сетей методом обучения без учителя для решения задач компьютерного зрения.
Дипломная работа / Минск: БГУ, 2020. – 44 с.

Рассматривается задача получения распутанных представлений методами обучения без учителя для дальнейшей классификации качества клубней картофеля.

АНАТАЦЫЯ

Гумілеўскі К.С. Працэс трэніроўкі штучных нейронных сетак метадам навучання без настаўніка для вырашэння задач камп’ютэрнага гледжання.
Дыпломная работа / Мінск: БДУ, 2020. – 44 с.

Разглядаецца задача атрымання расплутаных уяўленняў метадамі навучання без настаўніка для далейшай класіфікацыі якасці клубняў бульбы.

ABSTRACT

Gumilevski K.S. The process of training artificial neural networks in unsupervised manner for solving computer vision problems.
Diploma / Minsk: BSU, 2020. – 44 p.

It is considered the problem of obtaining disentangled representations using unsupervised algorithms for further classification of the quality of potato tubers.

РЕФЕРАТ

ПРОЦЕСС ТРЕНИРОВКИ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ МЕТОДОМ ОБУЧЕНИЯ БЕЗ УЧИТЕЛЯ ДЛЯ РЕШЕНИЯ ЗАДАЧ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Дипломная работа: 44 с., 37 рис., 3 табл., 12 источников

Ключевые слова: искусственные нейронные сети (ИНС), методы обучения без учителя, вариационный автокодировщик (VAE), распутанное представление.

Цель работы: получение распутанных представлений, разработка моделей и алгоритмов для классификации на основе распутанных представлений.

Методы исследования: общая теория ИНС, теория алгоритмов и моделей обучения без учителя.

Полученные результаты и их новизна: в работе получены распутанные представления для "реальных" данных, на их основе произведена классификация.

Область применения: алгоритмы получения распутанных представлений - для любых изображений, классификация качества клубней картофеля применима в сельском хозяйстве.

РЭФЕРАТ

ПРАЦЭС ТРЭНІРОЎКІ ШТУЧНЫХ НЕЙРОННЫХ СЕТАК МЕТАДАМ НАВУЧАННЯ БЕЗ НАСТАЎНІКА ДЛЯ ВЫРАШЭННЯ ЗАДАЧ КАМП'ЮТЭРНАГА ГЛЕДЖАННЯ

Ключавыя словы: штучные нейронные сеткі, метады навучэння без настаўніка, расплутанае уяўленне.

Мэта працы: атрыманне расплутаных уяўленняў, распрацоўка мадэляў і алгарытмаў для класіфікацыі на аснове расплутаных уяўленняў.

Атрыманыя вынікі і іх навізна: у рабоце атрыманы расплутаныя прадстаўленні для "рэальных" дадзеных, на іх аснове праведзена класіфікацыя.

Вобласць прымянення: алгарытмы атрымання расплутаных уяўленняў - для любых малюнкаў, класіфікацыя якасці клубняў бульбы - у сельскай гаспадарцы.

SUMMARY

THE PROCESS OF TRAINING ARTIFICIAL NEURAL NETWORKS IN UNSUPERVISED MANNER FOR SOLVING COMPUTER VISION PROBLEMS

Keywords: artificial neural networks(ANN), unsupervised methods, variational autoencoders (VAE), disentangled representation.

Objective: obtaining disentangled representations, developing models and algorithms for classification based on disentangled representations.

Research methods: general theory of ANN, theory of unsupervised algorithms.

The obtained results and their novelty: obtained disentangled representations for "real" data, made classification based on disentangled representations.

Scope: algorithms for obtaining disentangled representations are applicable to any image data, the classification of the quality of potato tubers is applicable in agriculture.

СОДЕРЖАНИЕ

Перечень сокращений и условных обозначений

ИНС – искусственные нейронные сети
КЛ-расстояние – расстояние Кульбака-Лейблера
СВ – случайная величина
ТИ – теория информации
IB-принцип – Information bottleneck принцип
VAE – вариационный автокодировщик
WAE – автокодировщик Вассерштайна
REC – ошибка реконструкции

ВВЕДЕНИЕ

Искусственные нейронные сети (ИНС) уже давно известны научному сообществу. Однако лишь недавно получили широкое практическое применение. Это стало возможным благодаря технологии CUDA, которая позволяет производить параллельные вычисления на GPU.

Выделяют два основных вида обучения ИНС: обучение с учителем и обучение без учителя. При использовании первого подхода ИНС обучается на размеченном наборе данных и предсказывает ответы, которые в дальнейшем используются для оценки точности алгоритма. При обучении без учителя модель использует неразмеченные данные, из которых алгоритм самостоятельно пытается извлечь признаки и зависимости. Сферы применения ИНС очень обширны: и машинный перевод, и распознавание образов и их применение в беспилотных автомобилях, и многое-многое другое. При этом большинство впечатляющих результатов получены с помощью алгоритмов обучения с учителем. Основной недостаток таких алгоритмов в том, что они требуют разметки большого количества данных. Отличительная особенность обучения без учителя заключается в том, что мы не передаём системе никакие знания о данных, модели сами находят признаки и внутренние зависимости в них.

В направлении обучения без учителя есть активно развивающаяся область обучения представлений. Данное направление предполагает, что все данные высокой размерности получены из малого количества признаков, так называемых факторов вариативности. Например, изображение лица может быть описано по цвету кожи, форме и цвету глаз, прическе и так далее. Каждое изображение это большой массив пикселей. Например, если изображение имеет размер 100 на 100, то оно описывается 100 * 100 = 10000 чисел. При этом каждое из этих чисел по отдельности не несёт в себе смысл. Направление получения распутанных представлений занимается получением сжатых представлений данных, в которых каждая размерность отвечает за какой-то свой, и притом единственный, признак. Такие представления могли бы использоваться для обучения на малом количестве примеров (подробнее идея описана в пункте 1.4) или для разделения данных по определенным признакам.

Получение распутанных представлений – активно развивающаяся область обучения без учителя, при этом в ней практически отсутствуют эксперименты на реальных данных. В данной работе описываются современные подходы для получения распутанных представлений и метрики их оценки. Основной акцент делается на получение распутанных представлений для изображений клубней картофеля. В качестве практического применения, такие представления используются для классификации качества клубней.

ГЛАВА 1
ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ И ТЕОРЕТИЧЕСКИЕ СВДЕНИЯ

1.1 Основные понятия и определения

Распутанное представление – сжатое представление данных, в котором каждая компонента отвечает ровно за один фактор вариативности данных. Например, если речь идёт об изображении клубня картофеля, то в распутанном представлении данного изображения ровно одна компонента должна отвечать за размер картофеля, ровно одна за форму и тд.

Автокодировщик (AE) – генеративная модель, которая учится отображать объекты в заданное скрытое (еще называют латентным или кодовым) пространство (как правило, значительно меньшей размерности, чем размерность объекта) и затем восстанавливать их обратно (реконструировать).

Реконструкция – данные, получаемые в результате декодирования сжатого представления исходных данных.

Гиперпараметры модели – параметры модели, которые задаются в начале обучения и не изменяются в течение обучения.

Компонента z_a сжатого представления VAE значимая, если $D_{KL}(q_\varphi(z_a|x)||N(0,1)) > 0.01$, где D_{KL} – расстояние Кульбака-Лейблера

Seed - число, используемое для инициализации генератора псевдослучайных чисел.

1.2 Расстояние Кульбака-Лейблера

Расстояние Кульбака-Лейблера (КЛ-расстояние) – неотрицательная, несимметричная мера удалённости друг от друга двух вероятностных распределений, определенных на некотором общем пространстве элементарных событий.

КЛ-расстояние распределения Q относительно P обозначается как $D_{KL}(P||Q)$. В ТИ КЛ-расстояние также интерпретируется как количество информации, потерянной при замене истинного распределения P на распределение Q.

В общем случае, если μ – любая мера на X, для которой существуют абсолютно непрерывные относительно μ функции p = $\frac{dP}{d\mu}$ и q = $\frac{dQ}{d\mu}$, тогда КЛ-расстояние распределения Q относительно P определяется как 1.1:

$$D_{KL}(P||Q) = \int_X p \log \frac{p}{q} d\mu \quad (1.1)$$

В данном случае основание логарифма существенной роли не играет. В зависимости от его значения КЛ-расстояние отличается с точностью до постоянного множителя, по сути, он только фиксирует единицу измерения. В случае, когда используется натуральный логарифм, КЛ-расстояние измеряется в натах, а когда двоичный – в битах. КЛ-расстояние – берзамерная величина вне зависимости от размерностей исходных распределений.

1.3 Основные виды обучения ИНС

Выделяют 2 основных вида обучения ИНС: обучением с учителем и обучение без учителя (рисунок 1.1).

При обучении с учителем имеется множество пар объект-ответ. Некоторой модели подаётся конечное количество таких пар, называемое обучающей выборкой. Требуется построить алгоритм, который сможет понять взаимосвязи между этими парами и научиться получать ответы для новых объектов (не из тренировочной выборки). При этом качество работы алгоритма оценивается по некоторому функционалу качеству. Как правило, функционал качества также использует для своего подсчета размеченные пары объект-ответ.

Алгоритму обучения без учителя на вход подаются только описания объектов (без ответов), при этом он сам должен найти внутренние взаимосвязи и зависимости в них.

По сути, основное отличие обучения без учителя от обучения с учителем заключается в том, что мы не передаём системе никакие знания о данных, модели обучения без учителя сами находят внутренние зависимости в них. Отсюда вытекает большое преимущество – отсутствие необходимости в разметке тренировочных данных (создании человеком пар объект-ответ).

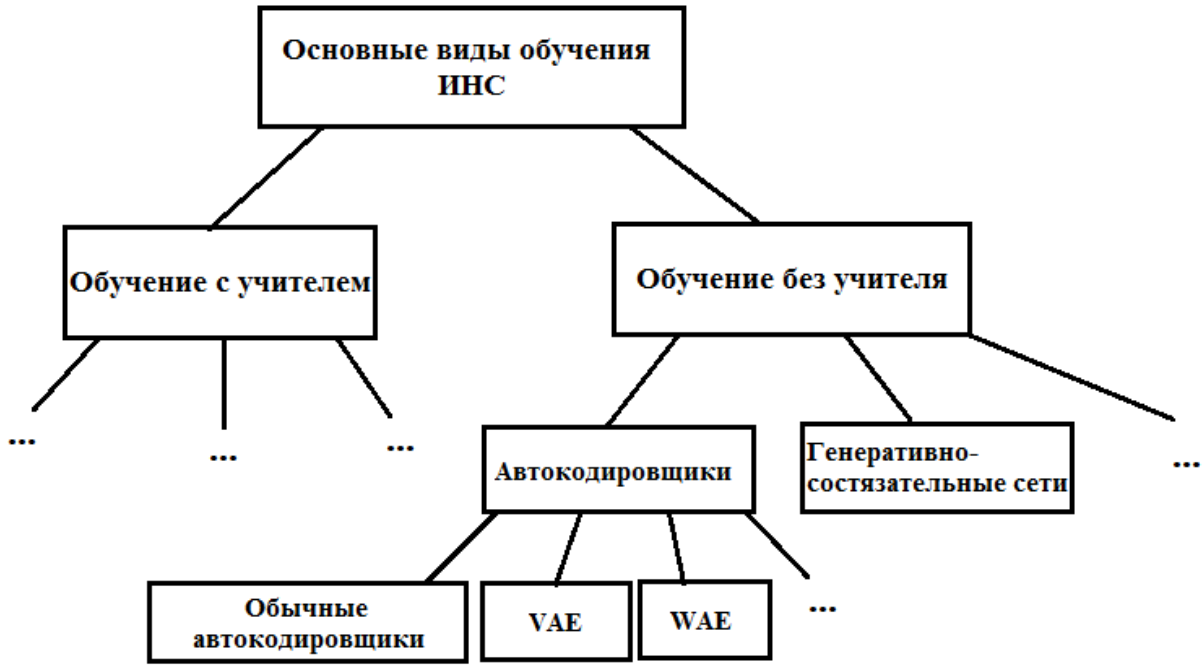

Рисунок 1.1. – Основные виды обучения ИНС.
(Тремя точками обозначены другие виды ИНС, с которые не рассматриваются в рамках данной работы.)

В данной работе основной акцент делается на алгоритмы обучения без учителя.

1.4 Обучение с учителем на малом количестве примеров

Для получения качественных результатов при обучении с учителем требуются десятки и даже сотни тысяч размеченных примеров. Это во многом

связано с большими размерностями входных данных. Чтобы работать с такими размерностями необходимо большое количество параметров в ИНС и, как следствие, большое количество размеченных примеров.

Эту проблему можно решить с помощью алгоритмов обучения без учителя. Моделям для получения распутанных представлений показывается большое количество неразмеченных данных. Они выделяют признаки в этих данных и хранят их в распутанных представлениях. Теперь для поставленной задачи используются не исходные примеры, а их распутанные представления. Такие представления имеют значительно меньшую размерность, а значит уже не требуют большого количества размеченных данных.

1.5 Теоретические сведения по теории информации

Зачастую ИНС анализируют с точки зрения теории информации ([8], [9]). Рассмотрим основные определения.

Пусть X – дискретная СВ.

Чем чаще наступает событие x_k, тем меньше информации оно в себе несёт. Частота наступления события определяется вероятностью p_k, поэтому количество информации $I(x_k)$ должно быть обратно пропорционально ему. В теории информации $I(x_k)$ определяется как $I(x_k) = log\frac{1}{p_k} = -logp_k$ (1.2). $I(x_k)$ также является дискретной СВ и её математическое ожидание называется энтропией H(X) СВ X(1.3). Энтропия H(X) является мерой неопределенности X.

$H(X) = E[I(x_k)] = \sum_k p_k\, I(x_k) = -\sum_k p_k \log p_k$ (1.3)

Пусть Y – дискретная СВ. Условной энтропией называют величину H(X|Y) = H(X, Y) – H(Y) (1.4). Она определяет количество неопределенности в X, которое осталось после наблюдения Y. H(X, Y) – совместная энтропия (1.5): $H(X, Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(x,y)$ (1.5)

Величина I(X, Y) = H(X) – H(X|Y) определяет количество неопределенности в X, которое было разрешено после наблюдения Y. I(X, Y) называют взаимной информацией (1.6):

$I(X, Y) = H(X) - H(X|Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log(\frac{p(x,y)}{p(x)p(y)})$ (1.6)

По аналогии с дискретным случаем вводятся понятия для непрерывных величин. Пусть X – непрерывная СВ с плотностью вероятности $f_X(x)$, Y – непрерывная СВ с плотностью вероятности $f_Y(y)$.

Энтропия $H(X) = -\int_{-\infty}^{+\infty} f_X(x)\, log f_X(x)dx$ = -E[$log f_X(x)$] (1.7)

Взаимная информация I(X, Y) (1.8):

$I(X, Y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f_{X,Y}(x,y)\log\left(\frac{f_X(x|y)}{f_X(x)}\right)dxdy = [f_{X,Y}(x,y) = f_X(x|y)\, f_Y(y)]$

$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f_{X,Y}(X,Y)\log(\frac{f_{X,Y}(x,y)}{f_X(x)f_Y(y)})dxdy$ (1.8),

где $f_{X,Y}(x,y)$ – плотность совместной вероятности X и Y, $f_X(x|y)$ – плотность условной вероятности X при Y = y.

1.6 IB-принцип

Рассмотрим некоторые данные X и задачу Y. Например, в случае с задачей классификации X – исходные данные, а Y – классы, к которым относится X.

Пусть Z – некоторое представление X. Рассмотрим задачу о максимизации взаимной информации между ответом Y и представлением Z при некоторых ограничениях на сложность данного представления. Ограничение на сложность может быть представлено в виде ограничения на взаимную информацию между X и Z. Таким образом, имеем следующую оптимизационную задачу: $\max I(Z,Y)$ при условии $I(X,Z) \leq I_c$, где I_c – некоторое информационное ограничение

С помощью правила множителей Лагранжа данная оптимизационная задача сводится к максимизации следующего функционала:

$$R_{IB} = I(Z,Y) - \beta I(Z,X)$$

Таким образом, наша цель – найти некоторое представление Z, которое максимально информативно об Y и в то же время достаточно сжато. Данный подход известен как Information bottleneck (IB-принцип) и был впервые предложен в 1999 году ([10]).

ГЛАВА 2. АЛГОРИТМЫ ОБУЧЕНИЯ БЕЗ УЧИТЕЛЯ ДЛЯ ПОЛУЧЕНИЯ РАСПУТАННЫХ ПРЕДСТАВЛЕНИЙ И МЕТРИКИ ОЦЕНКИ ИХ КАЧЕСТВА

2.1 Вариационный автокодировщик (VAE)

Вариационные автокодировщики [1] изначально были предложены для генерации новых данных из определённого распределения. Для генерации, VAE использует сжатое (латентное) представление z данных, которое, при определенных модификациях может получаться распутанным. Ключевое отличие VAE от других автокодировщиков заключается в том, что z принадлежат некоторому известному кодовому распределению $p_{model}(z)$, что значительно упрощает генерацию (мы заранее знаем из какого распределния генерировать), хотя и усложняет процесс обучения.

Для генерации выборки из модели, VAE сначала выбирает пример z из кодового распределния $p_{model}(z)$. Затем этот пример прогоняется через дифференцируемую генераторную сеть p_θ и производится выборка из распределения $p_\theta(x|z)$.

На этапе обучения для получения z (т.н. латентного представления) из x используется кодировщик (encoder) q_φ(z|x) и тогда $p_\theta(x|z)$ рассматривается как декодирующая сеть (декодер, decoder). Данные, получаемые на выходе декодера называют реконструкцией (рисунок 2.1).

Главная идея вариационных автокодировщиков заключается в том, что их можно обучить с помощью максимизации вариационной нижней границы (формула 2.1):

$$L_{VAE} = \sum_{i=1}^{n} \mathrm{E}_{z\sim q_\varphi(\mathrm{z}|x_i)} log p_\theta(x_i|z) - D_{KL}(q_\varphi(\mathrm{z}|x_i)||\, p_{model}(z)) \quad (2.1)$$

где $D_{KL}(||)$ – неотрицательное расстояние Кульбака-Лейблера между распределениями и n – количество объектов в тренировочных данных (рисунок 2.1).

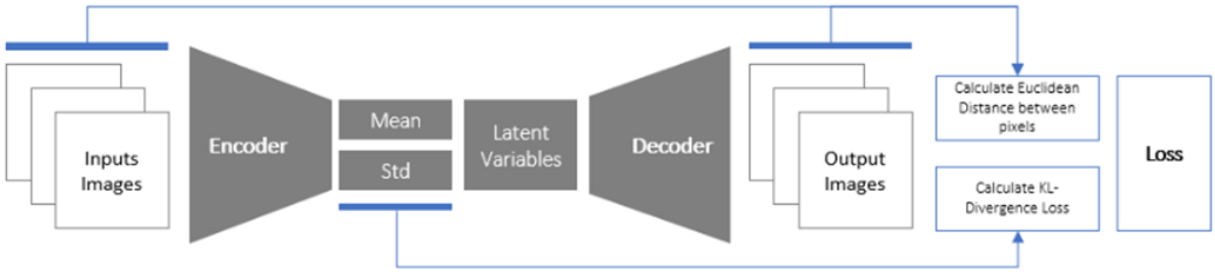

Рисунок 2.1. – Архитектура VAE на примере изображений (Latent Variables = z, mean и std – среднее и среднеквадратичное отклонение генерируемых кодировщиком распределний)

В первом слагаемом легко узнать логарифмическое правдоподобие реконструкции, встречающееся и в других автокодировщиках. На практике, оно обычно аппроксимируется среднеквадратичной разницей между входными данными и реконструкцией. Второе слагаемое (КЛ-слагаемое) сближает распределения $q_\varphi(\mathrm{z}|x)$ и $p_{model}(z)$. В VAE, $p_{model}(z)$ - стандартное нормальное

распределение N(0;1) и кодировщик принимает форму $q_\varphi(\mathrm{z}|x_i) = N(\mu(x_i), diag\ \sigma^2(x_i))$. Отметим, что в VAE ковариационная матрица диагональная, что играет важнейшую роль для получения распутанных представлений. Учитывая формукодировщика, второе слагаемое L_{VAE} может быть вычислено в приближенной форме как (формула 2.3):

$L_{KL} = \frac{1}{2}\sum_{j=1}^{d}(\mu_j^2(x_i) + \sigma_j^2(x_i) - log\sigma_j^2(x_i) - 1)$ (2.3), где d – размерность латентного пространства (2.3)

Также нормальное распределение позволяет использовать, так называемый, репараметризационный трюк: вместо распределения N(0;1) кодировщик q генерирует среднее μ_i и стандартное отклонение σ_i. Тогда $z_i = \mu_i + \sigma_i\epsilon$, где $\epsilon \sim N(0;\ 1)$ (Рисунок 2.2).

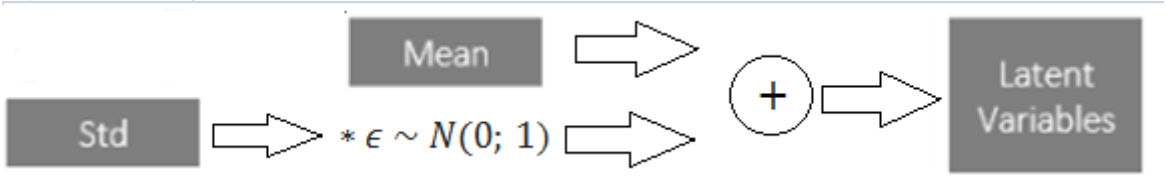

Рисунок 2.2. - Репараметризационный трюк в VAE

2.2 Модификации VAE для получения распутанных представлений

2.2.1 β −VAE

β −VAE [2] – это модификация VAE, которая добавляет гиперпараметр β в значение целевой функции L (формула 2.4):

$$\mathrm{L} = \sum_{i=1}^{N} \mathrm{E}_{z\sim q_\varphi(\mathrm{z}|x_i)} log p_\theta(x_i|z) - \beta D_{KL}(q_\varphi(\mathrm{z}|x_i)||\, p_{model}(z)) \quad (2.4)$$

Хорошо подобранное β (обычно $\beta > 1$) приводит к получению лучшей распутанности в латентном представлении z. Когда $\beta = 1$, β −VAE вырождается в VAE. Показывается, что большее ‘внимание’ на слагаемое с КЛ-расстоянием накладывает дополнительные ограничения на ёмкость латентного представления z, однако в то же время сохраняется возможность реконструировать данные.

2.2.2 Связь β −VAE с IB-принципом

Обычно IB-принцип формулируется для задач обучения с учителем, но в статье [8] показано, что он также тесно связан с функцией потерь β-VAE.

В ней рассматривается задача о максимизации взаимной информации между представлением Z данных X при накладывании ограничений на количество информации, которое может хранить Z об особенностях каждого из элементов из X (i) (формула 2.5):

$$\max I(Z, X) - \beta I(Z, i) \quad (2.5)$$

В статье показывается, что (2.5) сводится к функции потерь β-VAE. Таким образом, выход кодировщика q(z|x) может быть рассмотрен как информационный bottleneck (VAE-bottleneck) для задачи реконструкции данных $\sum_{i=1}^{N} \mathrm{E}_{z\sim q_\varphi(\mathrm{z}|x_i)} log p_\theta(x_i|z)$. Также можно говорить о накладывании

дополнительных ограничений на VAE-bottleneck при увеличении β. Такие выводы открывают новые перспективы для понимания β-VAE с точки зрения теории информации. Также удивляет, что функция потерь VAE(β-VAE) была получена из абсолютно других соображений, но имеет прямую связь с IB-принципом.

2.2.3 β −VAE и распутанные представления

За счет чего представлния получаются распутанными? Ключевая гипотеза из статьи [3] заключается в том, что β −VAE находит различные компоненты, которые вносят различный вклад в слагаемое реконструкции. Эти компоненты соответствуют некоторым признакам в данных. Почему эти признаки не “смешиваются” в сжатом представлении, а получаются распутанными? В той же статье выдвигается гипотеза (которая подтверждается практически), что β −VAE сначала “учит” признаки, наиболее влияющие на реконструкцию. После того, как они выучены, нам больше ‘невыгодно’, с точки зрения значения функции потерь, менять компоненты, сильно влияющие на слагаемое ошибки реконструкции, а значит новые признаки ‘записываются’ в новые компоненты z.

В 2019 году, в статье [11] эти предположения были доказаны теоретически. В [11] удалось показать, какие особенности в реализации VAE способствуют получению распутанных представлений. В частности, слагаемое ошибки реконструкции и диагональная ковариационная матрица, помогают декодеру находить ортоганальные направления в данных.

Было показано, что зачастую во время обучения VAE (β −VAE) попадает в режим, в котором многие компоненты не влияют на реконструкцию (такое поведение наблюдалось и в наших экспериментах). Распределение таких компонент совпадает с N(0;1). В статье показано, что когда обучение происходит в таком режиме, линейная аппроксимация декодера в точке x_i, $J_i = \frac{\partial Dec_\theta(\mu(x_i))}{\partial \mu(x_i)}$, стремится иметь ортогональные столбцы и разделять факторы вариативности на основе того, какой вклад в реконструкцию они несут. Такая ортогонализация вызывает получение распутанных латентных представлений. Именно поэтому кажется естественной модификация VAE, представленная в пункте 2.2.4.

2.2.4 β −TCVAE

В статье [7] рассматривается следующая декомпозиция КЛ-слагаемого VAE (формула 2.6), и анализируется, какие именно слагаемые декомпозиции наиболее способствуют получению распутанных представлений:

$$\mathbb{E}_{p(x_i)} D_{KL}\big(q(\mathrm{z}|x_i)||p(z)\big) =$$
$$D_{KL}(q(\mathrm{z}, x_i)||q(z)p(x_i)) + D_{KL}(q(\mathrm{z})||\prod_j q(z_j)) + \sum_j D_{KL}(q(z_j)||p(z_j)) \quad (2.6)$$

На рисунке 2.3 схематически представлена декомпозиция КЛ-слагаемого. Первое слагаемое декомпозиции – взаимная информация I(z, x_i) между входными данными x_i и их латентным представлением z (основанное на распределении

данных q(z, x_i)). В теории информации, второе слагаемое называют total correlation (TC) – одно из возможных обобщений взаимной информации на несколько переменных. И, наконец, третье слагаемое соответствует сумме КЛ-расстояний между распределениями отдельно взятой компоненты латентного представления $q(z_j)$ и кодовым распределением $p(z_j)$ (в случае с VAE, $p(z_j) = N(0;1)$).

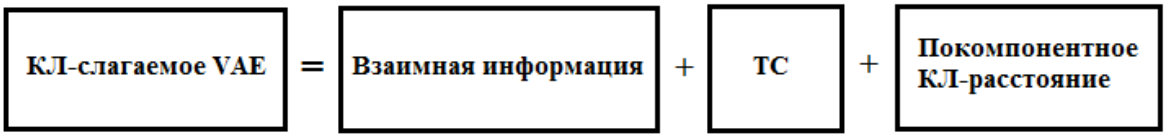

Рисунок 2.3. – Декомпозиция КЛ-слагаемого.

Первое слагаемое регулирует количество взаимной информации между входными данными и сжатым представлением. Накладывание дополнительных ограничений на данное слагаемое (например, в виде множителя большего 1) приводит к уменьшению взаимной информации между входом x и его сжатым представлением z. Кажется, что чем большее информации в представлении z об x, тем лучше это представление, поэтому авторы статьи пробовали убрать штраф на взаимную информацию (поставить множитель 0 перед первым слагаемым), однако это не дало улучшений. Это связано с тем, что на практике, даже при накладывании ограничений, данное слагаемое зачастую достигает своего максимума (log N). Таким образом, VAE "жертвует" большим значением взаимной информации для получения лучшей реконструкции.

Одно из важнейших свойств распутанных представлений - независимость признаков. Второе слагаемое способствует тому, чтобы VAE находил независимые факторы в данных. Кажется, что именно оно является наиболее важным для получения распутанных представлений (о чем свидетельствуют и теоретические результаты из [11]). И, наконец, третье слагаемое регулирует, чтобы каждая из компонент латентного представления была близка к N(0;1). Накладывание дополнительных ограничений на данное слагаемое приводит к тому, что все компоненты латентного представления становятся близки к стандартному нормальному распределению, что плохо по соображениям, описанным в пункте 2.2.5.

Таким образом, целевая функция β-TCVAE имеет вид (формула 2.7):

$$L_{\beta-\mathrm{TCVAE}} = \sum_{i=1}^{N} \mathrm{E}_{z\sim q_\varphi(\mathrm{z}|x_i)} log p_\theta(x_i|z) - D_{KL}(q(\mathrm{z},x_i)\big|\big|q(z)p(x_i)\big) - \beta D_{KL}(q(\mathrm{z})\big|\big|\prod_j q\big(z_j\big)\big) - \sum_j D_{KL}(q(z_j)||p(z_j)),\ \beta > 1 \quad (2.7)$$

2.2.5 Cβ −VAE

Как упоминалось ранее, на практике первое слагаемое зачастую аппроксимируется среднеквадратичной разницей и тогда минимизируется следующая функция потерь (формула 2.8):

$$L = ||x - \hat{x}|| + D_{KL}(q_\varphi(\mathrm{z}|x)||\, p_{model}(z)) \quad (2.8)$$

где $\hat{x} = p_\theta(x|z)$ – реконструкция

Один из недостатков β-VAE заключается в том, что оптимальное, с точки зрения функции потерь, состояние достигается, когда реконструкция $p_\theta(x_i|z)$ совпадает со входом x_i и $D_{KL}(q_\varphi(\mathrm{z}|x_i)||\, p_{model}(z))$ = 0 (минимум КЛ-расстояния достигается в 0). Исходя из декомпозиции, представленной в 2.2.4, понятно, что из $D_{KL}(q_\varphi(\mathrm{z}|x_i)||\, p_{model}(z))$ = 0 следует, что $\sum_j D_{KL}(q(z_j)||p(z_j))$ = 0, а значит кодировщик $q_\varphi(\mathrm{z}|x_i)$ переводит каждое x_i в сжатое представление z, такое, что каждая из компонент $z_i \sim p(z_i) = N(0;1)$.

Рассмотрим некоторый фактор вариативности и компоненту z_1, отвечающую за него. В условиях, когда $D_{KL} = 0$, кодировщик переводит любое изображение x_1 в z, т.ч. компонента z_1 ~ N(0; 1) и далее случайно выбирает из данного распределения. То есть независимо от наличия или значения фактора вариативности на изображении кодировщик всегда переводит изображения в одно и то же распределение N(0; 1).

На самом деле, мы хотим, чтобы в зависимости от наличия/значения некоторого фактора z_1 на изображении, оно переводилось в различные z_1 (в зависимости от значения фактора вариативности на изображении), то есть различные z_1-распределения.

Рассмотрим простой пример. Пусть компонента z_1 отвечает за такой фактор вариативности в изображении, как яркость. В таком случае мы хотим, чтобы светлые изображения отображались в одну область (распределение) z_1, а темные - в другую (рисунок 2.4).

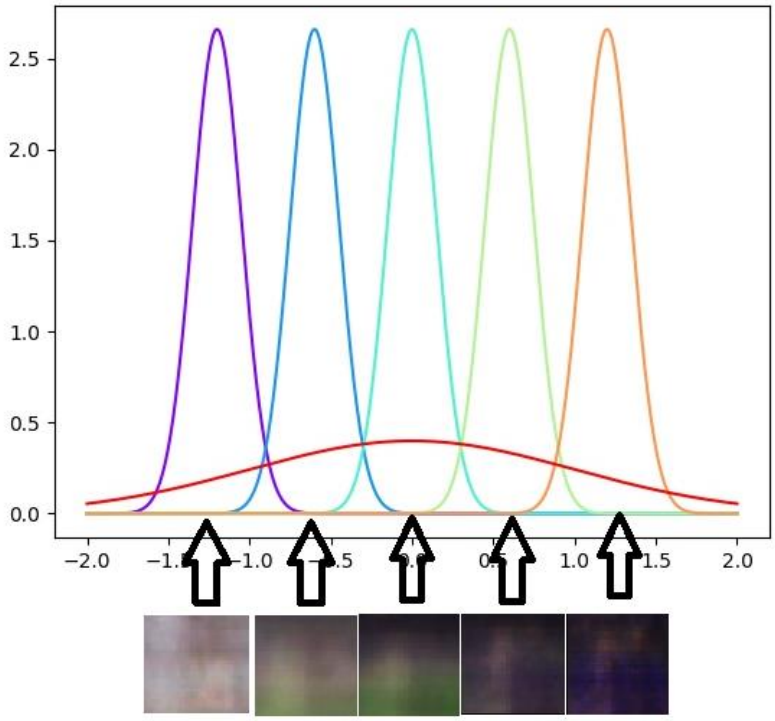

Рисунок 2.4. – Отображение кодировщиком изображений в различные распределения в зависимости от значения данного фактора вариативности (на примере фактора яркость).

Наличие таких областей означает перевод кодировщиком изображений в более узкие (со стандартным отклонением << 1) распределения с различными значениями среднего (зачастую отличными от 0). Оба эти фактора приводят к увеличению КЛ-расстояния $D_{KL}(q(z_j)||p(z_j))$, а значит и КЛ-слагаемого функции потерь, именно поэтому предлагается следующая модификация β-VAE – Cβ-VAE.

Cβ −VAE – модификация β −VAE, нацеленная на получение распутанных представлений. Впервые предложена в [3]. Целевая функция L для Cβ −VAE выглядит следующим образом (формула 2.5):

$$\mathrm{L} = \sum_{i=1}^{N} \mathrm{E}_{z\sim q_{\varphi}(\mathrm{z}|x_i)} log p_{\theta}(x_i|z) - \beta|D_{KL}\left(q_{\varphi}(\mathrm{z}|x_i)||p_{model}(z)\right) - C|,\ \mathrm{C} > 0 \quad (2.5)$$

В течение тренировки предлагается равномерно повышать значение C (например, на 0.5 каждые 50000 итераций обучения). КЛ-расстояние устремляется к C, тем самым искусственно повышается ёмкость z, что позволяет Cβ-VAE выделять новые компоненты. При этом хорошо подобранные C и β способствуют тому, что кодировщик фокусируется на выучивании какой-то одной компоненты в каждый момент тренировки.

2.3 Отслеживание динамики обучения и качественная оценка моделей

2.3.1 Метод варьирования компонент

Метод варьирования компонент (latent traversals) – это метод оценки распутанности, при котором смотрят на изменения в реконструкции при изменении некоторой фиксированной компоненты скрытого представления. Предполагается, что при хорошей распутанности изменение одной комоненты в латентном представлении будет приводить к изменению ровно одного фактора вариативности в изображении. На рисунке 4.1 представлено применение метода варьирования компонент для нашего набора данных.

Для получения рисунка, иллюстрирующего метод варьирования компонент, на вход подаётся некоторое изображение. Оно переводится кодировщиком в латентное представление z. В i-ом ряду меняется ровно одна компонента представления, от $z_i - 2$ до $z_i + 2$, остальные $z_j (j \neq i)$ фиксированы.

На рисунке 2.5 видно, что 0-ая и 6-ая компоненты меняют размер картофеля. Это плохо с точки зрения распутанности, так как сразу 2 компоненты отвечают за один и тот же фактор вариативности. 8-ая компонента меняет положение картофеля в кадре, а 1-ая − его форму (вытянутость). И, наконец, 2-ая компонента меняет яркость изображения, при этом с яркостью зачастую меняется и размер картофеля, так как он добавляет количество тёмных пикселей.

Стоит отметить, что оценка распутанности методом варьирования компонент значительно медленнее, чем вычисление некоторой метрики, и не даёт однозначный результат, какая из моделей лучше, когда это не заметно визуально.

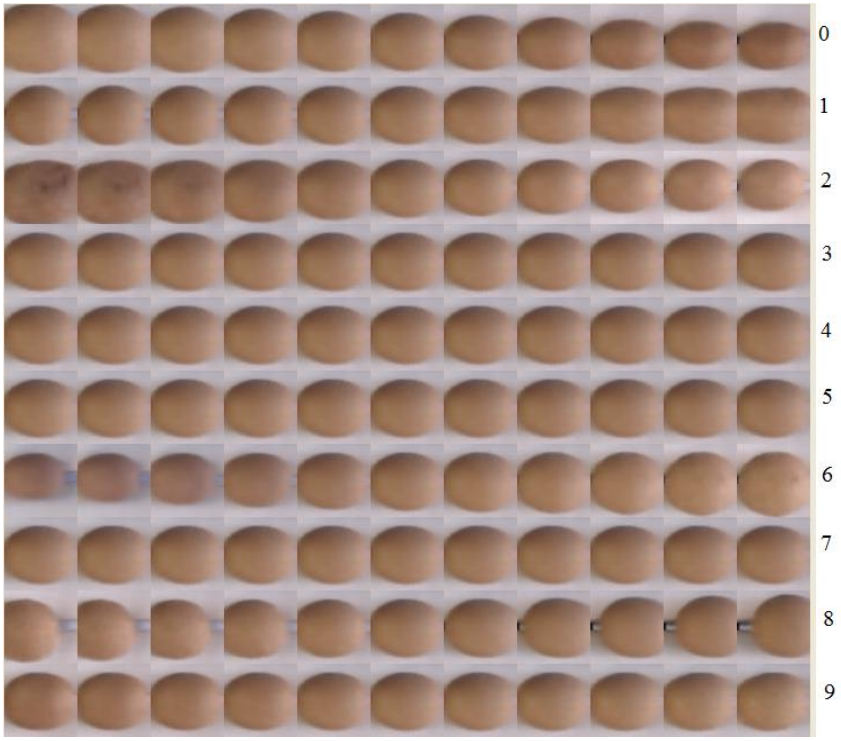

Рисунок 2.5. – Метод варьирования компонент для модели β-TCVAE ($\beta = 6,\ seed = 0$)

2.3.2 Отрисовка распределений компонент латентного представления

Диагональность ковариационной матрицы в VAE позволяет рассматривать каждую из компонент по отдельности. Для отслеживания динамики обучения и особенностей моделей использовалась отрисовка распределений компонент. Изображение x кодируется, в результате получается d значений среднего и дисперсии (где d – размерность латентного представления) и отрисовывается d нормальных распределений с полученными параметрами.

На рисунке 2.6 представлен пример отрисовки распределений для Cβ-VAE, при этом видно, что в данный момент обучения выделилось 2 значимые компоненты.

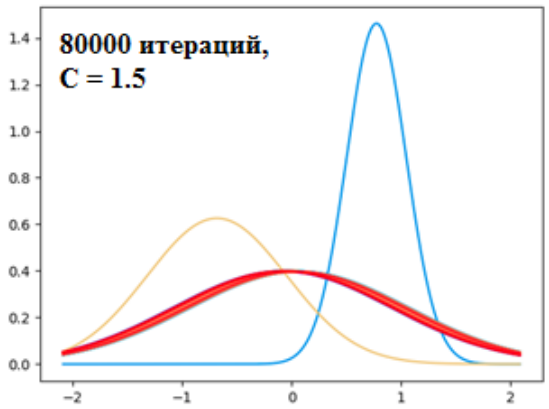

Рисунок 2.6 – Пример отрисовки компонент латентного представления для Cβ-VAE (на момент отрисовки C = 1.5, 80000 итераций обучения). Наблюдается 2 значимые компоненты.

2.3.3 UDR – оценка распутанности и сравнение моделей методом обучения без учителя

У метода варьирования компонент, как метода оценки распутанности, есть свои недостатки. Во-первых, он хуже подходит для других типов данных (например, таких, как аудио). Во-вторых, анализ полученных моделей с помощью метода варьирования компонент требует просмотра десятков и даже сотен изображений, что занимает очень много времени. И, наконец, главный недостаток – данный метод не дает однозначного ответа, какие модели лучше, когда это не заметно визуально. Поэтому я начал искать, какие метрики распутанности существуют в мире. В данном случае, под метрикой распутанности понимается некоторая мера, позволяющая получить численное значение того, насколько распутанные представления выдаёт модель. За последние годы было предложено множество метрик распутанности (Higgins [2], MIG [7] и др.). Главный недостаток этих метрик заключается в том, что они были придуманы для искусственно сгенерированных данных и используют при подсчете разметку по факторам вариативности.

В 2019 году, в статье [12], была предложена UDR метрика – первая метрика для качественной оценки распутанности методом обучения без учителя. Алгоритм вычисления (рисунок 2.7) следующий:

1) Обучается M = H * S моделей, где H – количество различных гиперпараметров моделей, и S – количество различных инициализаций весов (различных случайных seed'ов) моделей.
2) Для каждой из обученных моделей, выбирается (среди моделей с тем же значением гиперпараметра, что и у данной модели, но другим seed'ом) P $\leq$ S других обученных моделей.
3) Каждая из обученных моделей сравнивается с P выбранными моделями и вычисляется UDR_{ij} значение, где i $\epsilon \overline{1,\ M}$ j $\epsilon \overline{1,\ P}$
4) Конечный результат для i-ой модели UDR_i вычисляется как медиана (по j $\epsilon \overline{1,\ P}$) по значениям UDR_{ij}.

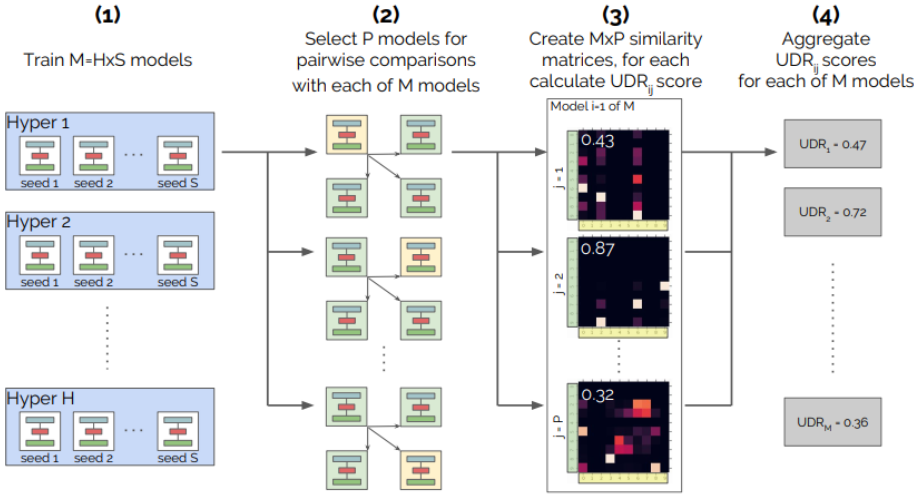

Рисунок 2.7. – Алгоритм вычисления UDR-метрики

UDR_{ij} показывает, насколько похожи латентные представления моделей i и j. Два латентных представления можно считать равными, если они совпадают с точностью до перестановки и знака (рисунок 2.8). Под совпадением с точностью

до перестановки подразумевается кодирование одних и тех же признаков (факторов вариативности), но под различными индексами в латентном представлении. А под совпадением с точностью до знака – кодирование одного и того же признака, но в обратном порядке.

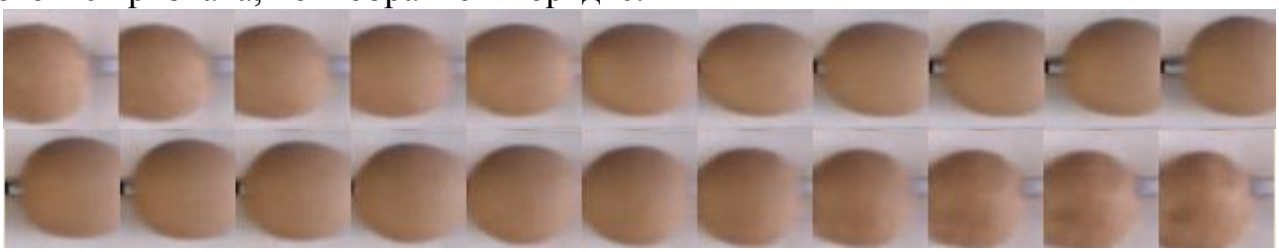

Рисунок 2.8. – Пример совпадения с точностью до знака: модели кодируют один и тот же признак (положение в кадре), но в разном порядке.

Эти 2 свойства учитываются при вычислении UDR_{ij}. Также стоит отметить, что UDR оценивает именно распутанность полученных компонент, поэтому при вычислении UDR_{ij} количество значимых компонент не учитывается.

Для учета совпадения с точностью до перестановки, при сравнении двух моделей рассматривается матрица R_{ij} размерности d x d (где d – размерность латентного представления). Клетка этой матрицы показывает схожесть i-ой компоненты латентного представления первой модели и j-ой компоненты второй. Для вычисления схожести, через кодировщик первой и второй моделей пропускается 1000 изображений, получаются латентные представления LR_1 и LR_2 (оба размерности 1000 x d). Из первого представления берем только i-ую компоненту - $LR_1[:, i]$, а из второго только j-ую - $LR_2[:, j]$. Между полученными векторами $LR_1[:, i]$ и $LR_2[:, j]$ (оба размерности 1000) считается метрика схожести – либо с помощью корреляции Спирмена (такой UDR называют UDR_S), либо с помощью регрессии Лассо (такой UDR называют UDR_L). Для учета совпадения с точностью до знака берется модуль матрицы $|R_{ij}|$. А для того, чтобы не учитывать количество значимых компонент, результат делится на сумму количества значимых компонент первой и второй моделей. Итоговая формула вычисления UDR_{ij} выглядит следующим образом (формула 2.6):

$$\frac{1}{d_a+d_b}\sum_b \frac{r_a^2 I_{KL}(b)}{\sum_a R(a,b)} + \sum_a \frac{r_b^2 I_{KL}(a)}{\sum_b R(a,b)} \quad (2.6)$$

где $r_a = max_a R(a,b)$, $r_b = max_b R(a,b)$. $I_{KL}(a)$ - индикатор того, является ли значимой компонента a, а $d_a = \sum_a I_{KL}(a)$ – количество значимых компонент первой модели и $d_b = \sum_b I_{KL}(b)$ – количество значимых компонент второй модели.

2.3.4 UDR: преимущества и недостатки

Безусловно, главное преимущество UDR в том, что эта метрика не требует никакой разметки. В то же время UDR основывается на сравнениях и может страдать от, так называемого, “стадного эффекта”. Из-за своей специфики некоторая модификация VAE может выучить отличное от других распутанное представление. Как результат, значение UDR будет занижено. И наоборот, наибольшее значение UDR может иметь модель, наиболее похожая на участвующие в сравнении (см. пункт 4.3.4). Именно поэтому нужно аккуратно

подходить к выбору моделей с помощью данной метрики. Эта особенность не касается подбора гиперпараметров, потому что при нем сравнения происходят в рамках одной модификации VAE и специфика модели уже не влияет на конечный результат. Поэтому UDR хорошо подходит для подбора гиперпараметров, что показано в пункте 4.3.3.

UDR метрика хорошо коррелирует с метриками распутанности, использующими разметку (рисунок 2.9).

SAMPLE # (P)	5	10	15	20	25	30	35	40	45
CORRELATION	0.51 ± 0.07	0.57 ± 0.03	0.57 ± 0.05	0.6 ± 0.03	0.59 ± 0.03	0.61 ± 0.02	0.61 ± 0.02	0.61 ± 0.01	0.61 ± 0.01

Рисунок 2.9. – Зависимость корреляции UDR с Higgins метрикой (использующей разметку по факторам вариативности) от количества сравнений P.

При этом чем больше сравнений P (см. 2ой шаг подсчета UDR) используется, тем лучше метрики коррелируют между собой. Отсюда появляется другой недостаток UDR – вычислительная затратность. Для подсчета метрики требуется обучать модель как минимум на P=5 различных случайных инициализациях весов. С одной стороны, это позволяет лучше обобщать получаемые результаты, но с другой – очень вычислительно дорого.

ГЛАВА 3. ПОСТАНОВКА ЗАДАЧИ И АЛГОРИТМ РЕШЕНИЯ

3.1 Постановка задачи

Дано:

RGB-изображения размера 640x480, на каждом из которых изображён один, вариативный по положению, размеру, цвету и тд, клубень картофеля.

Также дано разбиение данных на 2 класса: ‘хороший’ и ‘плохой’.

Требуется:

Проклассифицировать изображения: отнести каждое изображение к одному из классов: ‘хороший’ или ‘плохой’ клубень картофеля. То есть необходимо построить функцию $F: \mathbb{R}^{640x480x3} \rightarrow \{0, 1\}$, которая каждому изображению размера 640x480 сопоставляет класс.

В тренировочном датасете имеется 6944 изображений ‘плохих’ клубней картофеля (Рисунок 3.1) и 1554 ‘хороших’ (Рисунок 3.2).

Рисунок 3.1. – Примеры ‘плохих’ клубней

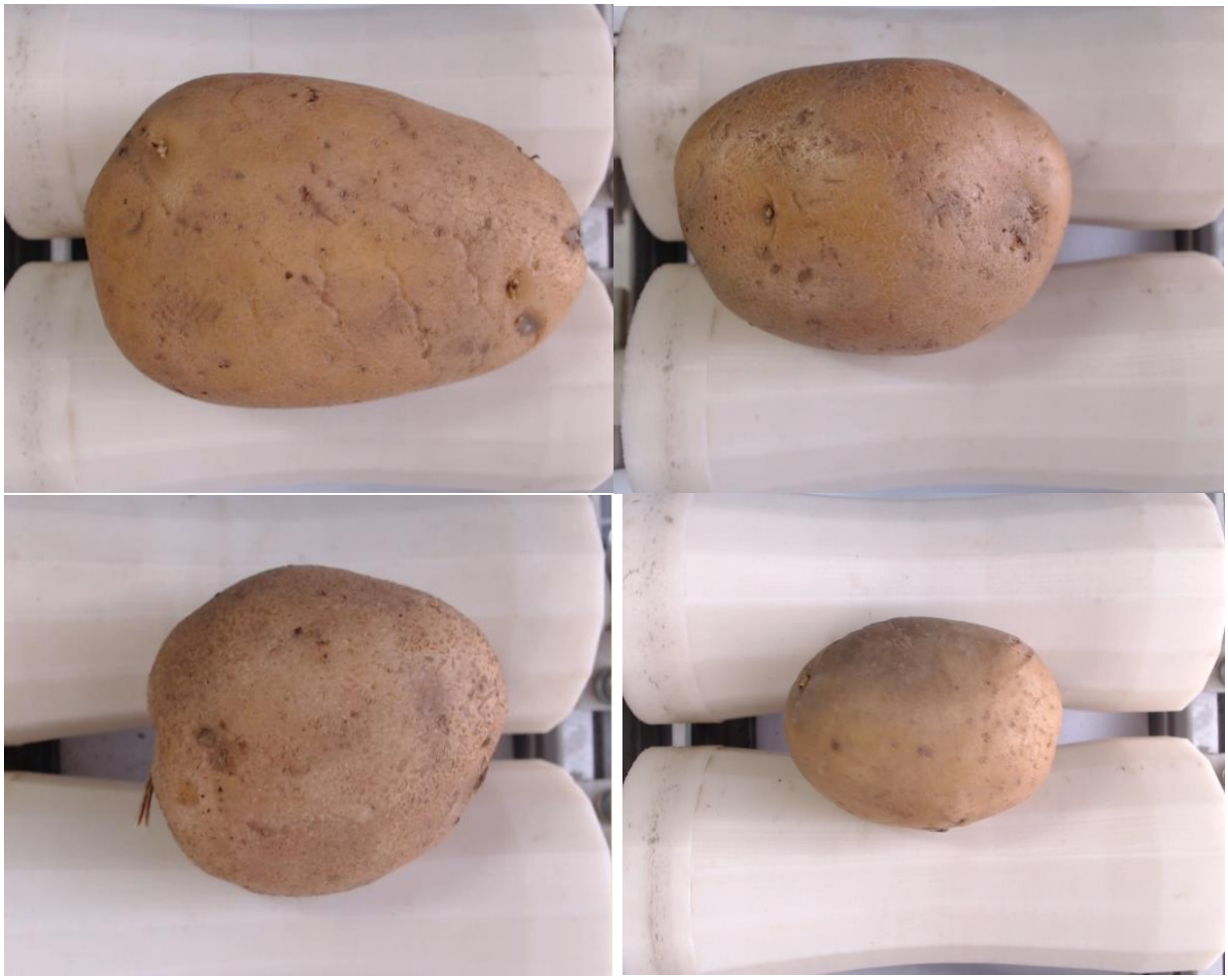

Рисунок 3.2. – Примеры ‘хороших’ клубней

3.2 Применение VAE к задаче классификации изображений клубней картофеля

На первом этапе (рисунок 3.3) обучаются различные модификации VAE для получения распутанных представлений: β-VAE, Cβ-VAE, β-TCVAE. Так как VAE относится к алгоритмам обучения без учителя, то на данном этапе разметка не требуется. У изображений предварительно берется центральная часть. В центральную часть, как правило, попадает клубень картофеля, а так как модификации VAE для получения распутанных представлений находят признаки, наиболее меняющие реконструкцию, то такая простая предобработка изображений позволяет получать больше признаков именно для клубней картофеля (а не, например, фона). Изображение сжимается до размера 64 на 64. Каждая из моделей обучается на различном наборе параметров (для того, чтобы выбрать наилучшую конфигурацию модель-гиперпараметры). Также для лучшей обобщаемости и подсчета UDR каждый набор гиперпараметров обучается на 5 различных инициализациях весов (различных seed’ах).

На втором этапе вычисляется UDR и на его основе выбираются лучшие модели. Выбранные модели анализируются с помощью инструментов, описанных в пунктах 2.3.1, 2.3.2 (метод варьирования компонент, отрисовка покомпонентных распределений и анализ реконструкции). На последнем этапе отбираются

релевантные для определения качества картофеля признаки и на их основе производится классификация. При этом для определения порогов для 1-2 релевантных признаков (использовался простейший алгоритм классификации: в зависимости от значения релевантного признака по порогу определялся класс) требуется небольшая разметка. В работе использовалось 200 размеченных изображений.

Отбор релевантных признаков и дальнейшую классификацию можно заменить некоторым алгоритмом машинного обучения. К примеру, можно применить обучение на малом количестве примеров, описанное в пункте 1.4, и ИНС сама отберет релевантные признаки и проклассифицирует по ним. Или же можно использовать алгоритмы из классического машинного обучения: случайный лес, XGBoost и другие. Но этот вопрос не был исследован в рамках данной работы, так как основной акцент делается на получение распутанных представлений с помощью алгоритмов обучения без учителя.

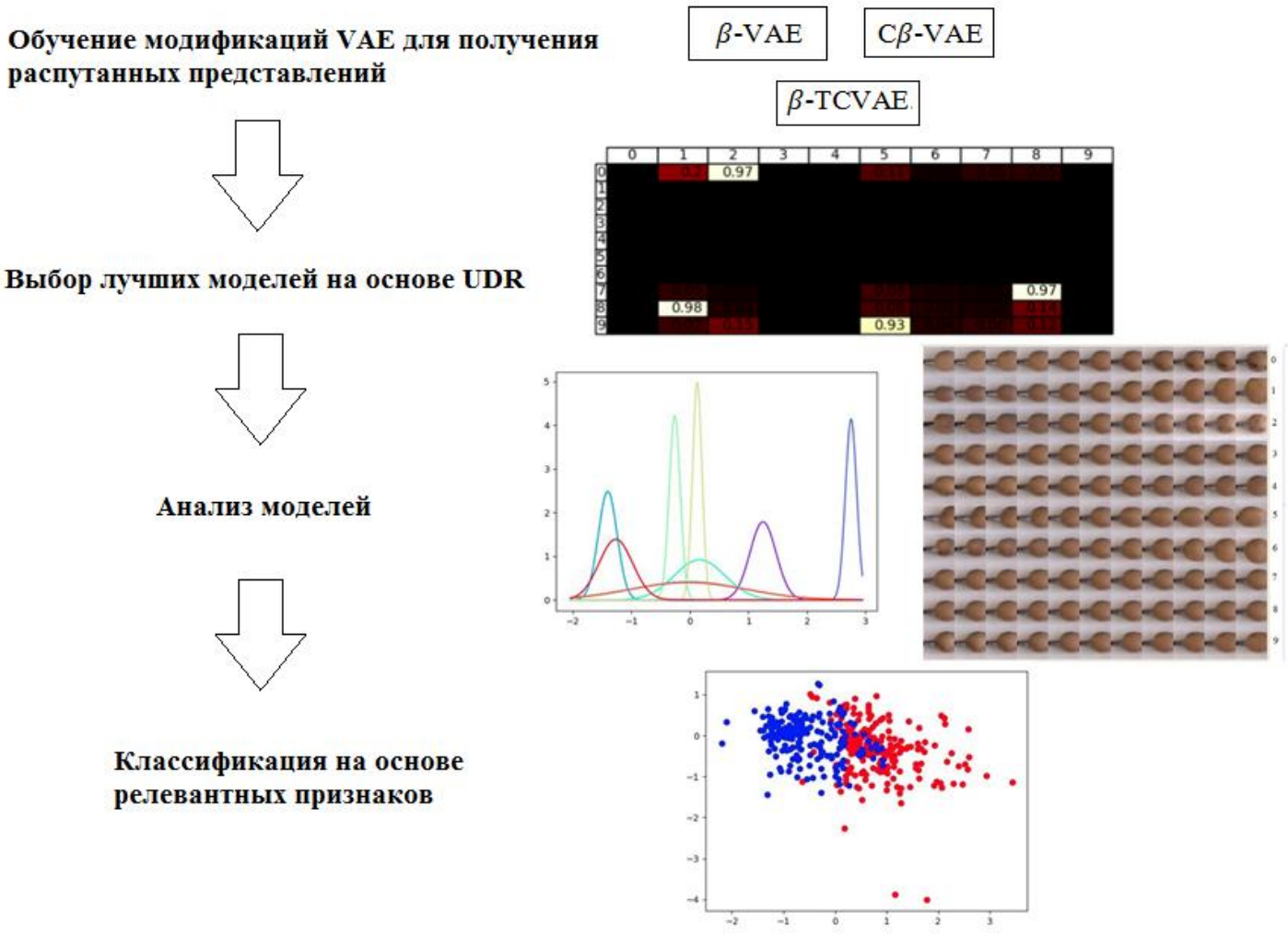

Рисунок 3.3. – Классификация на основе распутанных представлений

ГЛАВА 4. РАЗРАБОТКА ПРОГРАММНОГО ИНСТРУМЕНТАРИЯ И ЭКСПЕРИМЕНТЫ

4.1 Использовавшиеся технологии.

При написании моделей использовался язык Python. Python отлично подходит для исследовательских проектов в сфере машинного обучения. Для него написано множество различных полезных бибилиотек и фреймворков: тут и фреймфорки для разработки в сфере глубоко обучения, различные библиотеки для работы с данными, для визуализации и т.д. Всё это значительно ускоряет процесс разработки. В то же время Python довольно легко освоить, а код получается очень читабельным, что также немаловажно.

Теперь давайте рассмотрим более детально использование некоторых библиотек и фреймворков.

Зачастую глубокие ИНС представляются в виде графа (потока) вычислений. Под фреймворком глубокого обучения понимается фреймворк, который умеет определять этот граф, дифференцировать и вычислять его. При этом чем быстрее фреймворк умеет выполнять данные операции, тем лучше. Сейчас, когда существует технология CUDA, все фреймворки умеют использовать видеокарты на полную мощь, а значит операции дифференцирования и вычисления графа выполняются быстро. На первое место стал вопрос гибкости определения графа вычислений.

Фреймворки глубокого обучения можно разделить на 2 категории: фреймворки со статическим графом вычислений и фреймворки с динамическим графом вычислений. В фреймворках со статическим графом вычислений можно создать граф любого размера и сложности, при этом после компиляции останутся доступны лишь операции прямого и обратного прохода графа. С одной стороны, это ускоряет вычисления, но с другой - усложняет отладку. К представителям данной категории относятся фреймворки MXNet, Theano и TensorFlow. В фреймворках с динамическим графом ввычислений можно, по сути, перестраивать граф перед каждым его запуском. В таких фреймворках граф строится динамически при каждом прямом проходе. Такой подход даёт максимальную гибкость и упрощает процесс разработки. Представители: Torch, PyTorch.

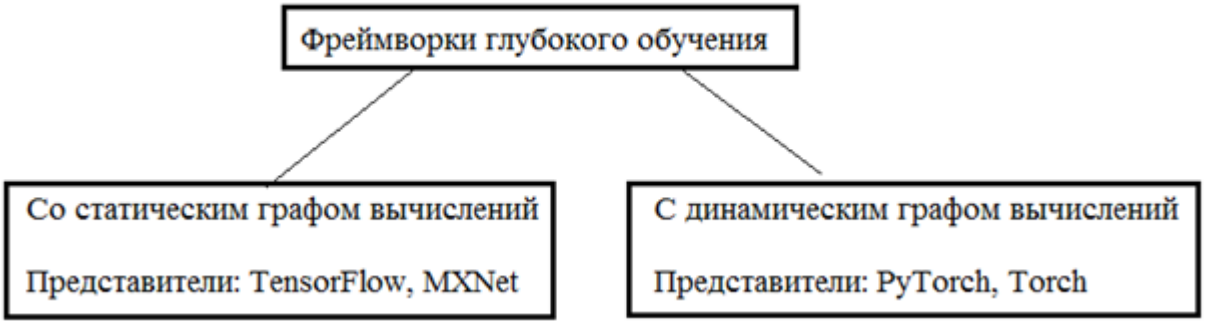

Рисунок 4.1. – Классификация фреймворков глубокого обучения по типу графа вычислений

В качестве основного фреймворка для построения и обучения моделей глубокого обучения был выбран PyTorch ([5], [6]). Для загрузки данных использовался пакет PyTorch utils. Также для подготовки и предобработки данных использовался пакет PyTorch torchvision. Torchvision содержит в себе некоторые стандартные модели для работы с компьютерным зрением и различные возможности для предобработки изображений (torchvision.transforms). Для визуализации архитектур нейронных сетей иногда полезно использовать TensorBoard, который, в том числе, доступен для PyTorch. Для построения графиков и других визуализаций использовалась библиотека Python matplotlib.

4.2 Эксперименты

4.2.1 Обучение моделей и анализ их особенностей с помощью отрисовки распределений компонент латентного представления

В рамках проведенных экспериментов было обучено 115 моделей для получения распутанных представлений. Из них 45 β-VAE: 9 различных гиперпараметров ($\beta = 1, 2, 3, 4, 6, 8, 10, 14, 18$) на 5 различных инициализациях весов (seed'ax). 40 моделей β-TCVAE: 8 гиперпараметров ($\beta = 2, 3, 4, 6, 8, 10, 14, 18$) на 5 seed'ax. И 30 моделей Cβ-VAE: 2 различных значения C_{max}= 6, 12 (в течение обучения C равномерно повышалось от 0 до C_{max}), 3 значения β = 700, 1400, 2100 (β подбирается так, что КЛ-расстояние сходится достаточно близко к C) на 5 seed'ax.

Как было упомянуто ранее, анализ динамики обучения проводился с помощью отрисовки распределений компонент латентного представления. Данный метод представляет теоретический интерес и во многом объясняют качество реконструкции получаемых моделей.

Каждая из обученных моделей имеет свои особенности. Например, Cβ-VAE выделяет компоненты последовательно, начиная от наиболее меняющих реконструкцию и заканчивая наименее. Например, у Cβ-VAE (β=700, C=6, seed=0) первая выделилась 0-ая компонента (рисунок 4.2), отвечающая за яркость (яркость пикселей наиболее меняет реконструкцию). Вторая выделилась 8-ая компонента, отвечающая за положение картофеля в кадре. Затем выделились 7-ая и 9ая компоненты, отвечающие за форму и размер соответсвенно.

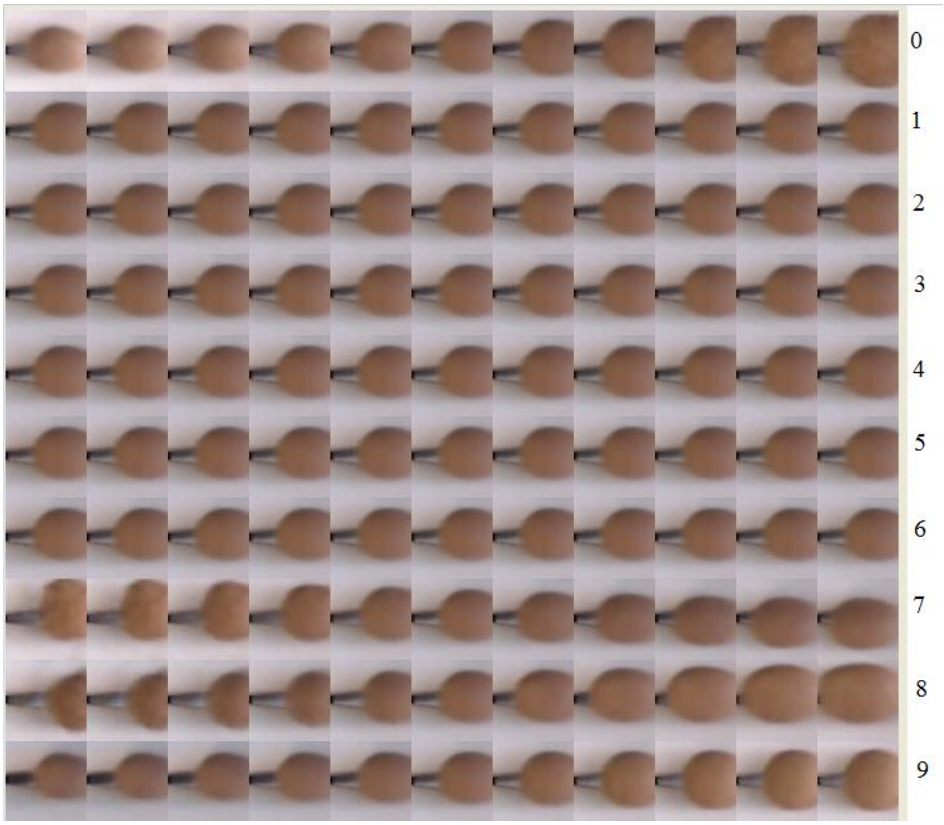

Рисунок 4.2. – Метод варьирования компонент для Cβ-VAE (β = 700, C = 6, seed = 0)

Выделение новых компонент зачастую соответствует повышению C. Это поведение можно объяснить с точки зрения декомпозиции КЛ-слагаемого, представленного в пункте 2.2.5. При повышении C “разрешается” пропускать больше информации через VAE-bottleneck и повышается значение допустимого КЛ-расстояния. Как следствие, выделяются новые компоненты (рисунок 4.3) или же “вытягиваются” распределения старых (рисунок 4.4). “Вытягивание” компонент может быть связано с тем, что при данных C и β, Cβ-VAE больше не может выделить признаки в исходных данных.

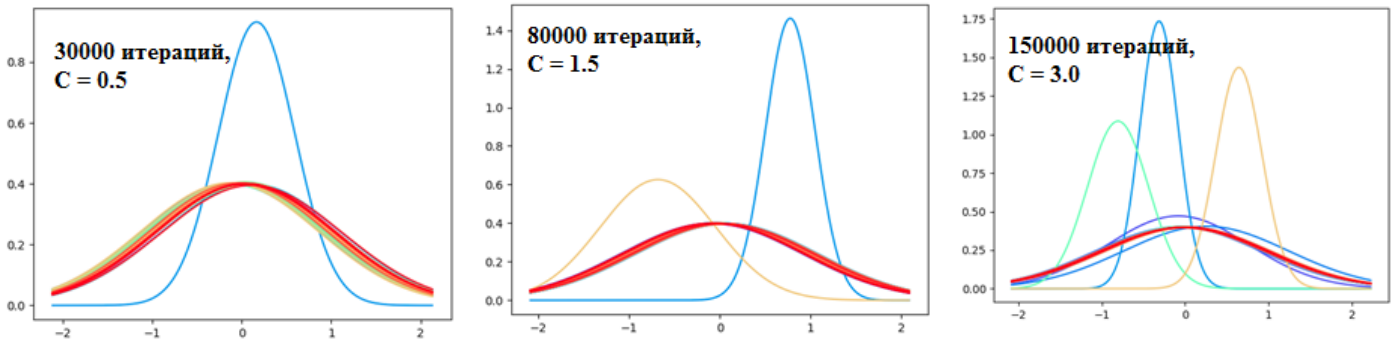

Рисунок 4.3. – Последовательное выделение значимых компонент в Cβ-VAE

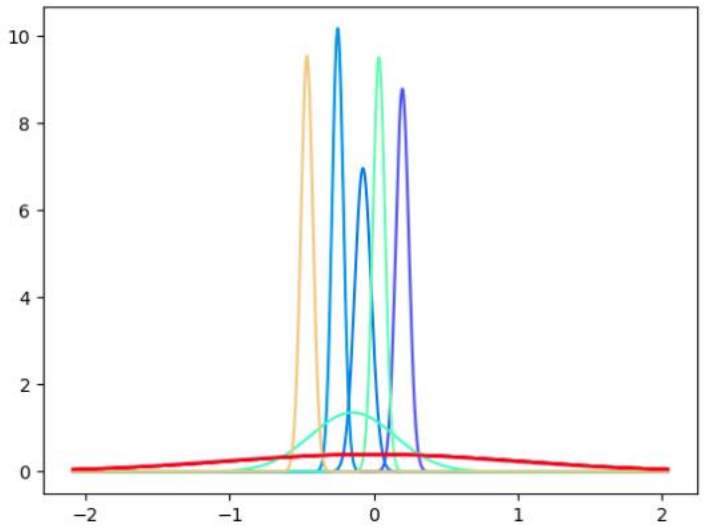

Рисунок 4.4. – Пример ‘вытигивания’ распределений компонент

β-TCVAE накладывает меньшие ограничения на взаимную информацию и покомпонентное КЛ-расстояние ($L_{\beta-\text{TCVAE}} = REC + KL_{\text{взаимная информация}} + \beta KL_{TC} + KL_{\text{покомпонентный}}$). Как следствие, распределения компонент получаются более узкими и с разнообразными средними. Исходя из рассуждений, приведенных в пункте 2.2.5, понятно, что это приводит к лучшей реконструкции. VAE и β-TCVAE накладывают одинаковые ограничения на покомпонентное КЛ-расстояние ($L_{VAE} = REC + KL_{\text{взаимная информация}} + KL_{TC} + KL_{\text{покомпонентный}}$), поэтому их распределения (рисунок 4.5, рисунок 4.6) и реконструкции (рисунок 4.7, рисунок 4.8) выглядят похожим образом, при этом у β-TCVAE значительно лучше распутанность получаемых представлений (рисунок 4.10). У β-TCVAE можно легко понять значения компонент: 0 – добавляет черные пятна, 1 – размер, 2 – яркость, 5 – положение, 6 – размер, 9 – форма. А вот у обычного VAE (рисунок 4.9) значительно сложнее выделить однозначный смысл компонент, при этом многие из них несут похожий смысл, что также свидетельствует о плохой распутанности.

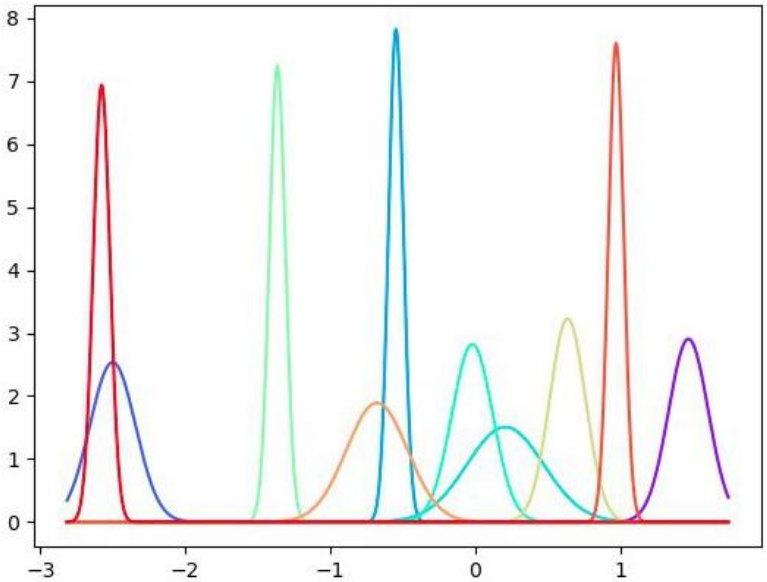

Рисунок 4.5. – Распределения компонент в VAE (seed = 0) (сверху)

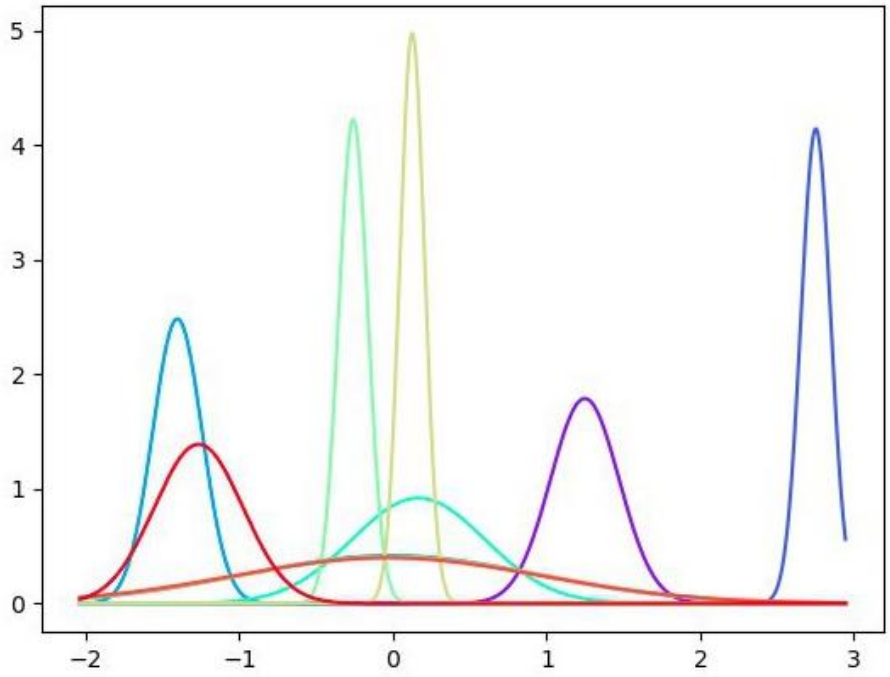

Рисунок 4.6. – Распределения компонент в β-TCVAE (β = 4, seed = 0). Распределения похожи на распределения VAE (рисунок 4.5) по своей вытянутости и разнообразию средних, у β-TCVAE меньше значимых компонент.

Рисунок 4.7. – Реконструкция VAE (seed = 0). Значение ошибки реконструкции 7627 (чем меньше, тем лучше).

Рисунок 4.8. – Реконструкция β-TCVAE (β = 4, seed = 0) (снизу). Значение ошибки реконструкции 7648 (чем ниже, тем лучше).

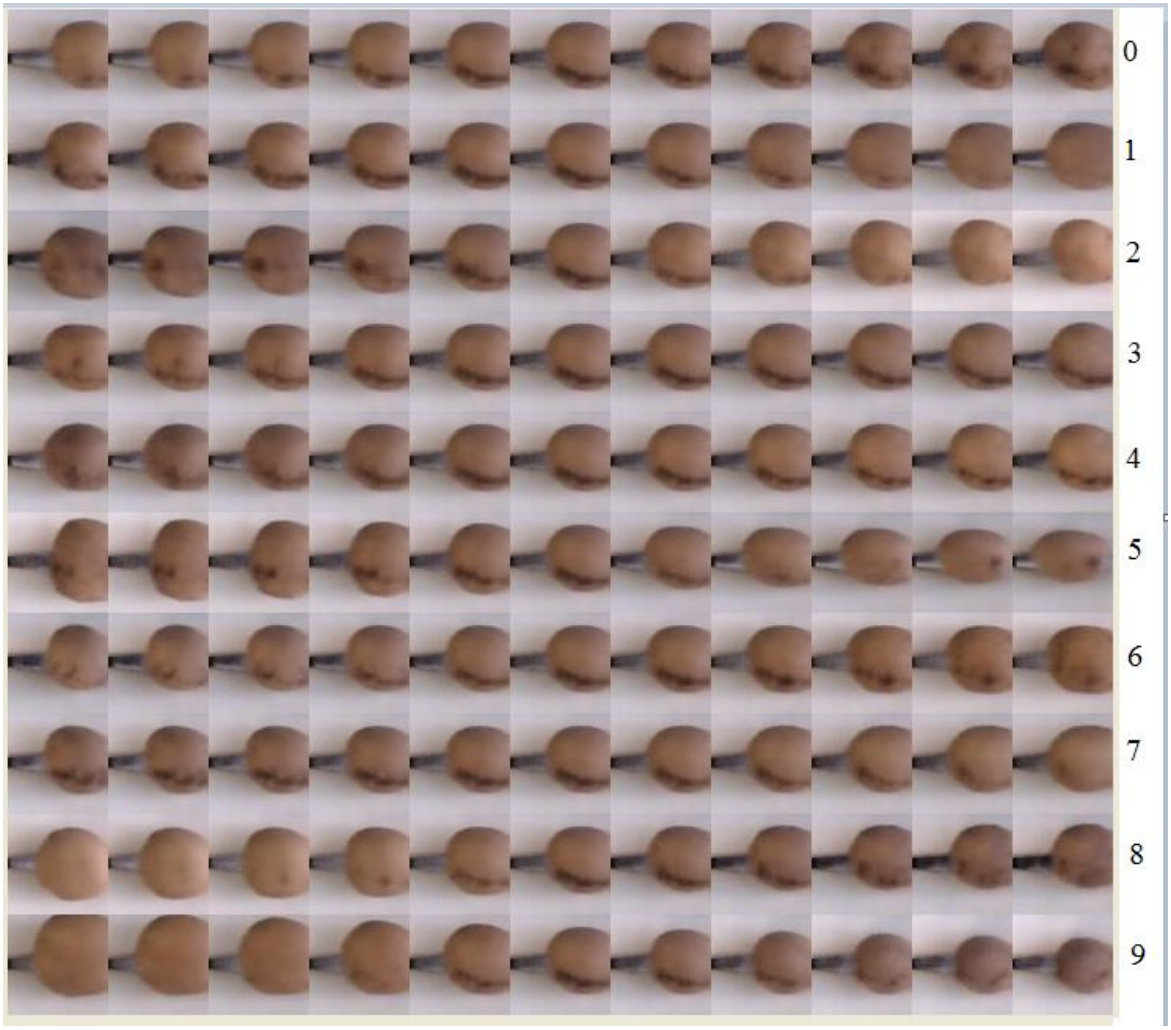

Рисунок 4.9. - Метод варьирования компонент для VAE (seed = 0)

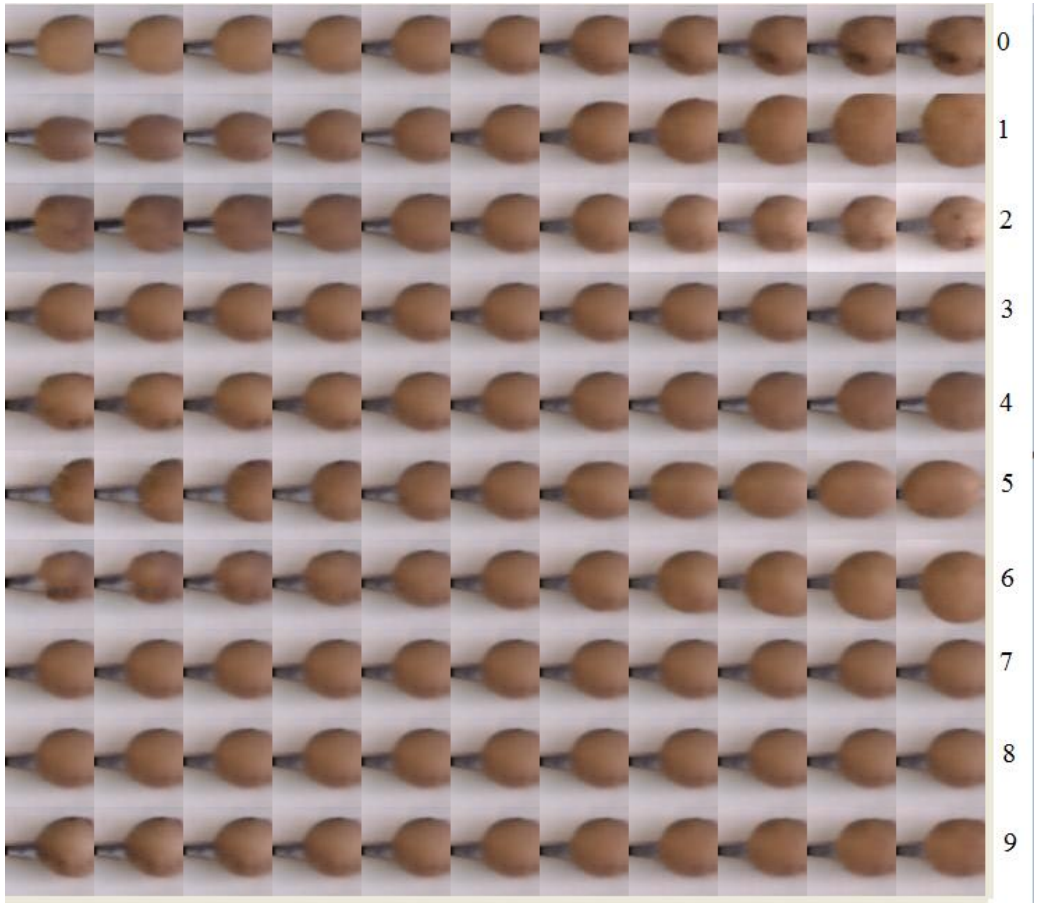

Рисунок 4.10. - Метод варьирования компонент дл β-TCVAE (β = 4, seed = 0).

4.2.2 Сравнение и выбор моделей на основе UDR

В данном пункте представлены 4 различных таблицы с UDR значениями. UDR-A2A (рисунок 4.11) - при вычислении UDR в сравнении участвуют все модели. И 3 UDR'а для каждой из моделей (β-VAE – рисунок 4.12, Cβ-VAE – рисунок 4.13, β-TCVAE – рисунок 4.14) - в сравнении участвует данная модель при различных гиперпараметрах и seed'ах.

Каждая из таблиц имеет следующую структуру: в первом столбце указан тип модели, во втором – значение гиперпараметра (в случае с Cβ-VAE через нижнее подчеркивание указано значение β и C), в третьем – количество значимых компонент, в последующих пяти столбцах значение UDR на различных seed'ах, и, наконец, последний столбец, UDR, – медиана по значениям предыдущих 5 столбцов.

	model type	param	#top	seed_0	seed_1	seed_2	seed_3	seed_4	udr
0	vae	1	10	0.25	0.30	0.30	0.32	0.28	0.30
1	beta_vae	2	10	0.36	0.35	0.35	0.36	0.38	0.36
2	beta_vae	3	8	0.40	0.36	0.35	0.35	0.36	0.36
3	beta_vae	4	7	0.46	0.42	0.37	0.33	0.34	0.37
4	beta_vae	6	6	0.45	0.50	0.44	0.49	0.45	0.45
5	beta_vae	8	6	0.46	0.39	0.49	0.44	0.49	0.46
6	beta_vae	10	6	0.45	0.47	0.44	0.43	0.42	0.44
7	beta_vae	14	4	0.46	0.47	0.46	0.46	0.45	0.46
8	beta_vae	18	4	0.45	0.45	0.45	0.41	0.46	0.45
9	beta_tc_vae	2	9	0.29	0.34	0.28	0.39	0.30	0.30
10	beta_tc_vae	3	8	0.33	0.39	0.37	0.37	0.36	0.37
11	beta_tc_vae	4	7	0.37	0.37	0.35	0.41	0.38	0.37
12	beta_tc_vae	6	6	0.38	0.39	0.39	0.42	0.47	0.39
13	beta_tc_vae	8	6	0.42	0.40	0.40	0.42	0.48	0.42
14	beta_tc_vae	10	6	0.39	0.42	0.39	0.50	0.53	0.42
15	beta_tc_vae	14	5	0.38	0.46	0.42	0.48	0.49	0.46
16	beta_tc_vae	18	5	0.46	0.42	0.48	0.46	0.46	0.46
17	c_beta_vae	700_6	4	0.47	0.48	0.47	0.46	0.46	0.47
18	c_beta_vae	700_12	7	0.33	0.44	0.38	0.46	0.45	0.44
19	c_beta_vae	1400_6	4	0.44	0.39	0.38	0.47	0.37	0.39
20	c_beta_vae	1400_12	4	0.36	0.44	0.42	0.38	0.44	0.42
21	c_beta_vae	2100_6	4	0.39	0.40	0.42	0.42	0.42	0.42
22	c_beta_vae	2100_12	4	0.32	0.33	0.35	0.35	0.35	0.35

Рисунок 4.11. – UDR-A2A – при вычислении UDR участвуют все модели.

	model type	param	#top	seed_0	seed_1	seed_2	seed_3	seed_4	udr
0	vae	1	10	0.28	0.30	0.31	0.33	0.29	0.30
1	beta_vae	2	10	0.35	0.37	0.36	0.38	0.38	0.37
2	beta_vae	3	8	0.38	0.38	0.40	0.37	0.39	0.38
3	beta_vae	4	7	0.48	0.45	0.39	0.35	0.38	0.39
4	beta_vae	6	6	0.44	0.51	0.46	0.53	0.44	0.46
5	beta_vae	8	6	0.50	0.40	0.50	0.48	0.50	0.50
6	beta_vae	10	6	0.53	0.52	0.46	0.44	0.43	0.46
7	beta_vae	14	4	0.38	0.48	0.47	0.46	0.45	0.46
8	beta_vae	18	4	0.36	0.46	0.46	0.36	0.46	0.46

Рисунок 4.12. – UDR β-VAE

	model type	param	#top	seed_0	seed_1	seed_2	seed_3	seed_4	udr
0	c_beta_vae	700_6	4	0.62	0.59	0.63	0.63	0.65	0.63
1	c_beta_vae	700_12	7	0.42	0.54	0.50	0.54	0.65	0.54
2	c_beta_vae	1400_6	4	0.64	0.56	0.53	0.63	0.52	0.56
3	c_beta_vae	1400_12	4	0.42	0.62	0.57	0.43	0.59	0.57
4	c_beta_vae	2100_6	4	0.52	0.51	0.59	0.61	0.59	0.59
5	c_beta_vae	2100_12	4	0.44	0.42	0.45	0.50	0.46	0.45

Рисунок 4.13. - UDR Cβ-VAE

	model type	param	#top	seed_0	seed_1	seed_2	seed_3	seed_4	udr
0	beta_tc_vae	2	9	0.39	0.41	0.33	0.43	0.39	0.39
1	beta_tc_vae	3	8	0.50	0.45	0.44	0.45	0.44	0.45
2	beta_tc_vae	4	7	0.51	0.44	0.50	0.50	0.51	0.50
3	beta_tc_vae	6	6	0.52	0.42	0.48	0.51	0.47	0.48
4	beta_tc_vae	8	6	0.49	0.48	0.49	0.51	0.53	0.49
5	beta_tc_vae	10	6	0.44	0.48	0.50	0.52	0.52	0.50
6	beta_tc_vae	14	5	0.44	0.37	0.46	0.42	0.43	0.43
7	beta_tc_vae	18	5	0.47	0.32	0.41	0.34	0.34	0.34

Рисунок 4.14. – UDR β-TCVAE

Стоит отметить, что UDR не учитывает количество значимых компонент, а только их распутанность. Поэтому надо понимать, что на моделях с большими значениями UDR, но меньшим количеством значимых компонент могут быть “утеряны” важные для классификации картофеля признаки. Например, понятно, что положение картофеля в кадре меняет значительно больше пикселей в изображении (сильнее влияет на реконструкцию), чем, например, черные пятна на картофеле. Поэтому второй признак скорее будет “утерян”, чем первый. При этом первый признак не является релевантным для определения качества картофеля, а второй – является.

То есть в некоторых случаях мы можем пренебречь более плохой распутанностью для сохранения важных для классификации признаков. Больше всего нас интересуют модели с наибольшими значениями распутанности при наибольшем количестве значимых компонент.

Как уже упоминалось ранее, UDR основывается на сравнениях представлений и предположении, что распутанные представления схожи между собой, а не распутанные – нет. На рисунке 4.15 представлена матрица корреляций UDR_{ij} для двух хороших, с точки зрения распутанности, моделей.

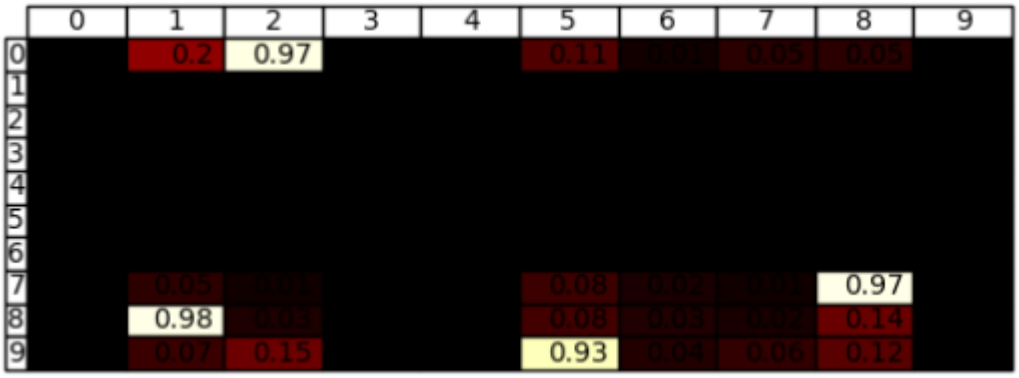

Рисунок 4.15. – матрица корреляций UDR_{ij}, сравниваются представления моделей Cβ-VAE ($\beta = 700$, C = 6, seed = 0) и β-VAE ($\beta = 8$, seed = 0)

Данная матрица показывает, насколько “похожи” (коррелируют) латентные представления двух моделей (см. пункт 2.3.3). В столбец записаны номера компонент первой модели, а в строку – второй. Полностью черная строка означает, что данная компонента первой модели незначимая. В клетке матрицы UDR_{ij} записывается значения корреляции между i-ой компонентой первой модели и j-ой компонентой второй. На рисунке 4.15 наблюдается 4 значимые компоненты, при этом 0-ая компонента первой модели сильно коррелирует со 2ой компонентой второй модели, а также 7ая первой с 8ой второй, 8ая первой с 1ой второй и 9ая первой с 5ой второй.

Это подтверждается и на методе варьирования компонент (рисунок 4.16). 0-ая компонента первой модели и 2-ая второй меняют яркость изображения. При этом зачастую (как в случае с первой моделью) с яркостью меняется и размер картофеля, так как он добавляет количество тёмных пикселей. 8-ая компонента первой модели и 1-ая компонента второй модели меняют положение картофеля, а 7-ая компонента первой модели и 8-ая компонента второй модели меняют его

форму. И, наконец, 9-ая компонента первой модели и 5-ая компонента второй меняют размер картофеля.

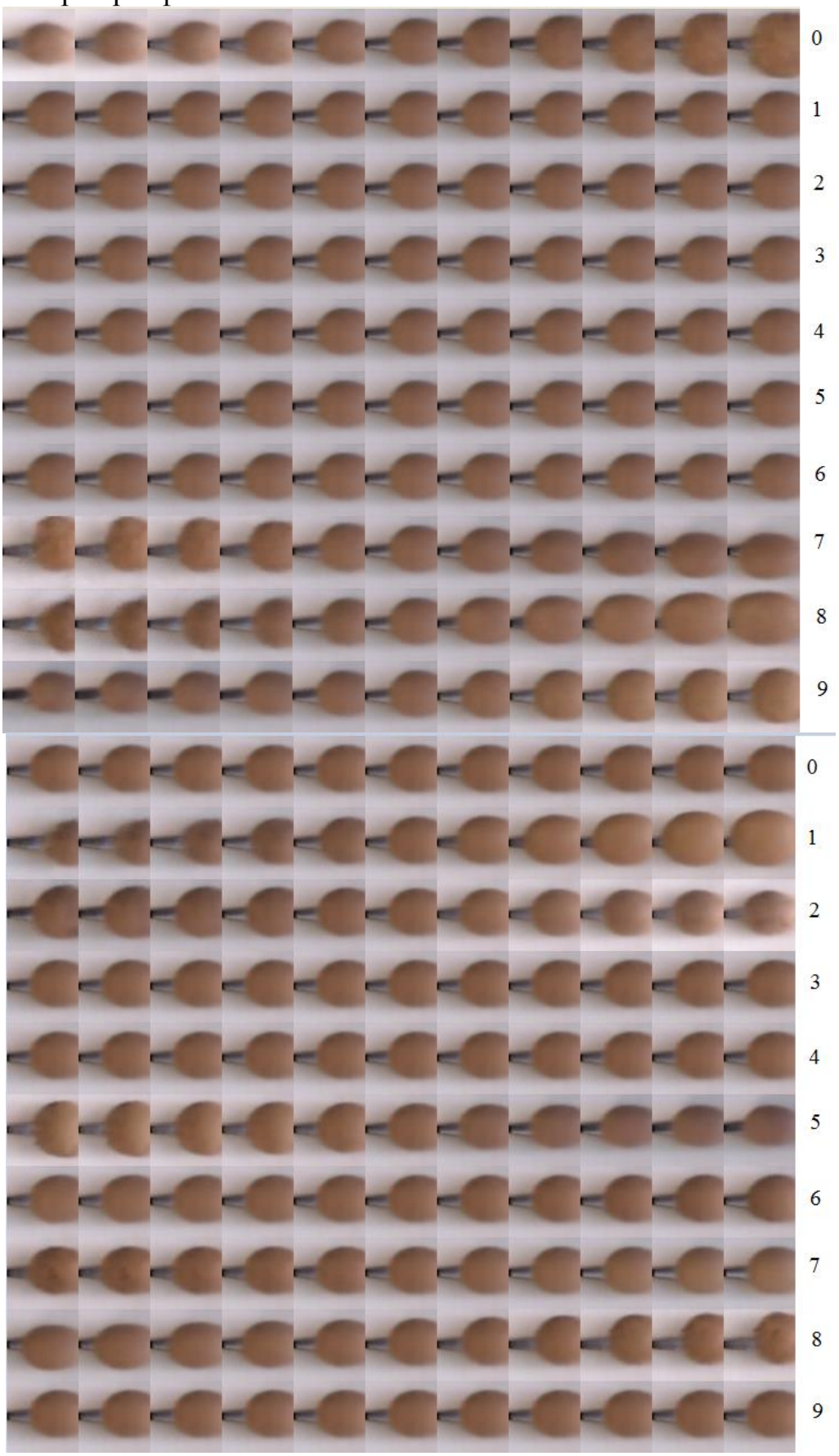

Рисунок 4.16. - Метод варьирования компонент для Cβ-VAE, сверху ($\beta = 700$, C = 6, seed = 0) и β-VAE, снизу ($\beta = 8$, seed = 0)

На рисунке 4.17 представлена матрица корреляций UDR_{ij} для плохой, с точки зрения распутанности, модели VAE (seed = 0) и того же Cβ-VAE ($\beta = 700$, C = 6, seed = 0). По ней видно, что все компоненты VAE значимые, при этом почти каждая компонента коррелирует сразу со многими компонентами Cβ-VAE.

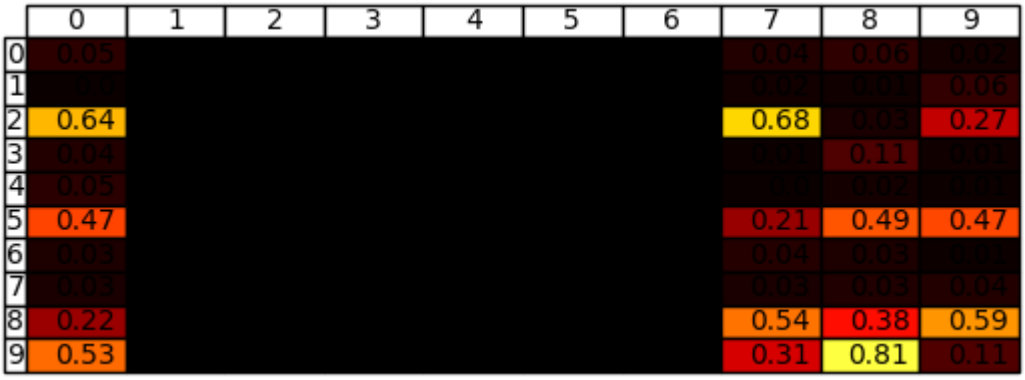

Рисунок 4.17. – матрица корреляций UDR_{ij}, сравниваются представления моделей VAE (seed = 0) и Cβ-VAE ($\beta = 700$, C = 6, seed = 0)

Плохая распутанность VAE подтверждается и на методе варьирования компонент (рисунок 4.9, верхняя картинка).

4.3.4 Подбор гиперпараметров с помощью UDR и “стадный” эффект UDR

С помощью UDR можно выбирать гиперпараметры моделей, на которых получается наилучшая распутанность. Рассмотрим на примере гиперпараметра β модели β-TCVAE.

	model type	param	#top	seed_0	seed_1	seed_2	seed_3	seed_4	udr
0	beta_tc_vae	2	9	0.39	0.41	0.33	0.43	0.39	0.39
1	beta_tc_vae	3	8	0.50	0.45	0.44	0.45	0.44	0.45
2	beta_tc_vae	4	7	0.51	0.44	0.50	0.50	0.51	0.50
3	beta_tc_vae	6	6	0.52	0.42	0.48	0.51	0.47	0.48
4	beta_tc_vae	8	6	0.49	0.48	0.49	0.51	0.53	0.49
5	beta_tc_vae	10	6	0.44	0.48	0.50	0.52	0.52	0.50
6	beta_tc_vae	14	5	0.44	0.37	0.46	0.42	0.43	0.43
7	beta_tc_vae	18	5	0.47	0.32	0.41	0.34	0.34	0.34

Рисунок 4.18. – UDR β-TCVAE

UDR β-TCVAE (рисунок 4.18) говорит о том, что одна из лучших моделей среди β-TCVAE, с точки зрения распутанности, - β-TCVAE с β=4 и seed=4. Одни из худших - β-TCVAE с β=2 и seed=2 и β-TCVAE с β=18 и seed=1. Проверим это на методе варьирования компонент. На рисунке 4.19 представлен метод варьирования компонент для β-TCVAE с β=2 и seed=2, на 4.20 – для β-TCVAE с β=4 и seed=4, на рисунке 4.21 – для β-TCVAE с β=18 и seed=1.

Нетрудно заметить, что у модели, представленной на рисунке 4.19 довольно плохая распутанность: например, 5ая компонента добавляет черные пятна, но также меняет и размер (а это различные факторы вариативности). Помимо 5ой, и 4-ая и 7-ая компоненты меняют размер (а в распутанных представлениях каждая компонента должна отвечать за ровно один признак). 7-ая компонента также, как

и 5-ая, добавляет пятна, а 4-ая помимо размера меняет цвет картофеля. Также сразу несколько компонент (2, 4, 5, 9) меняют цвет картофеля.

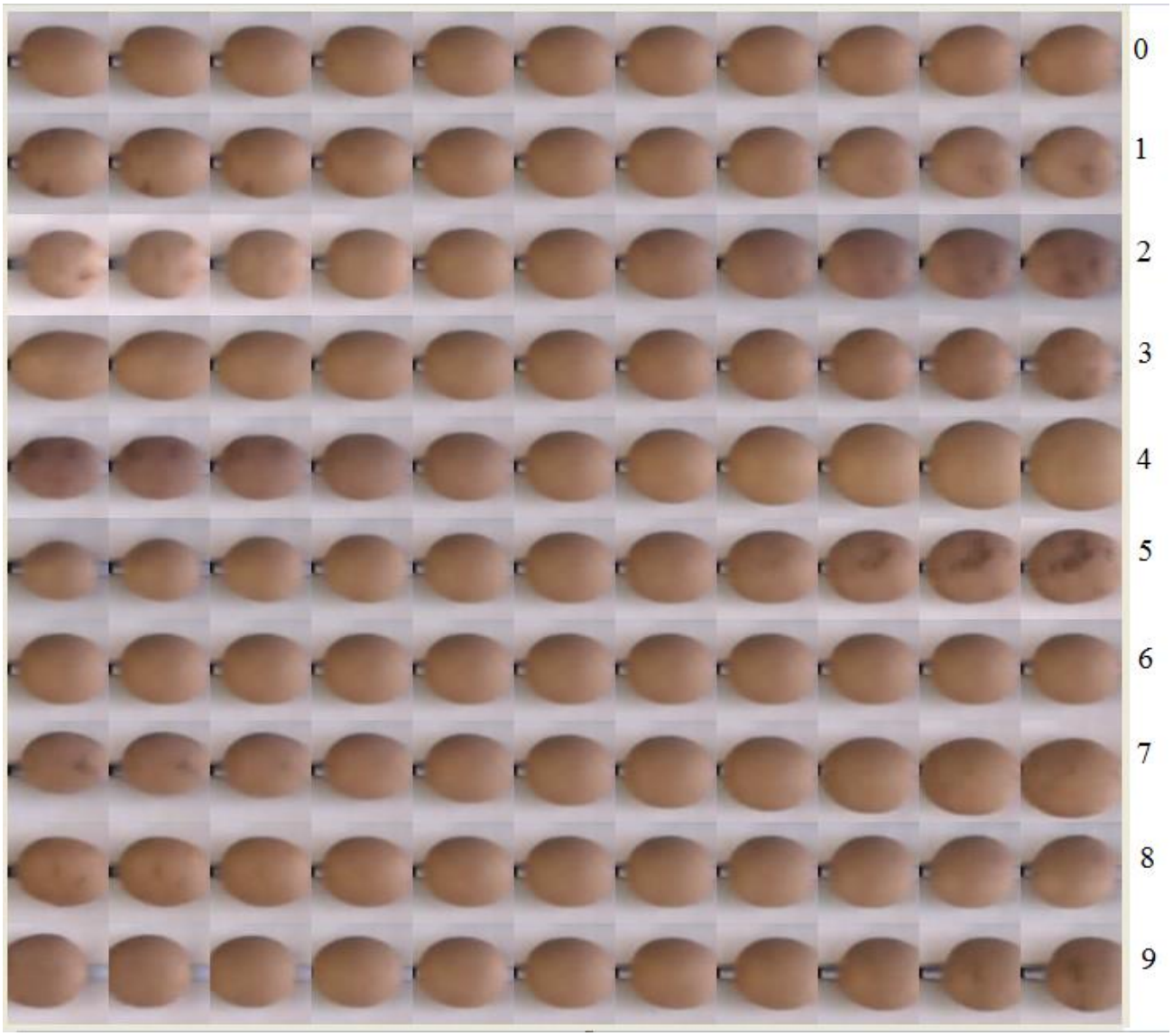

Рисунок 4.19. – Метод варьирования компонент для β-TCVAE с β=2 и seed=2.

На рисунке 4.20 видно, что распутанность значительно лучше. Да, у данной модели тоже есть недостатки, например, 1-ая и 6-ая компоненты ведут себе похоже. При этом ровно одна компонента (5-ая) меняет положение картофеля в кадре, ровно одна (9-ая) – форму (вытянутость) картофеля и так далее.

Несмотря на то, что модель, представленная на рисунке 4.21, имеет довольно мало значимых компонент, она все равно довольно плохая с точки зрения распутанности, о чем говорит и её UDR значение. У β-TCVAE с β=18 и seed=1 7-ая компонента отвечает за яркость, при этом меняется и размер картофеля (такой недостаток характерен большинству моделей, ведь увеличение размера добавляет тёмные/светыле пиксели). 3-ая компонента меняет и цвет, и положение и размер, а 4-ая и форму, и цвет.

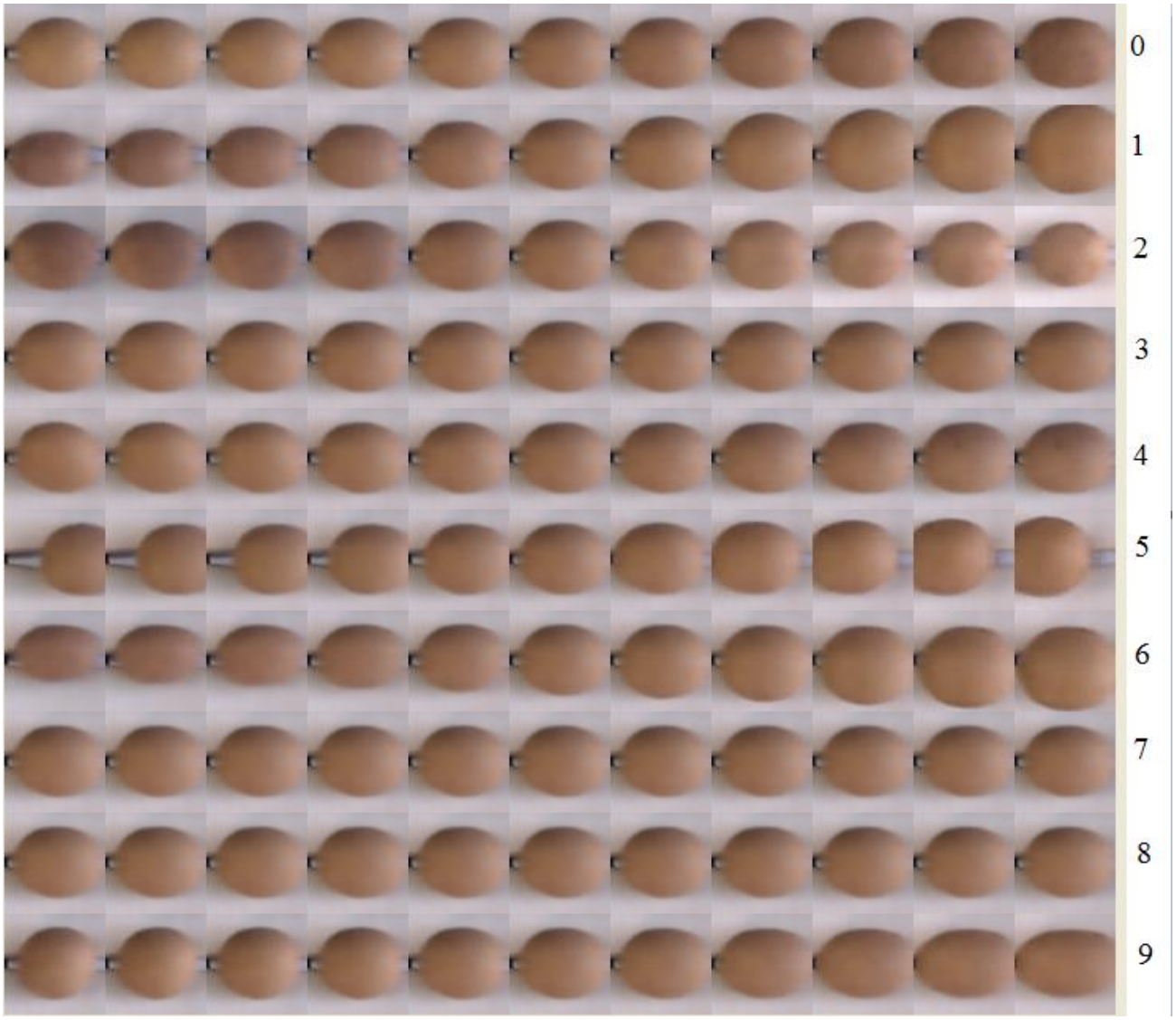

Рисунок 4.20. - Метод варьирования компонент для β-TCVAE с β=4 и seed=4.

Данный пример также показывает, что UDR может страдать от, так называемого, “стадного эффекта”, когда в сравнении участвуют все модели. “Стадный эффект” заключается в том, что из-за своей специфики некоторая модификация может выучить отличное от других распутанное представление (или же больше признаков). Как результат, значение UDR данной модели будет занижено. И наоборот, наибольшее значение UDR может иметь модель, наиболее “похожая” на участвующие в сравнении. На рисунке 4.11 видно, что в UDR-A2A β-TCVAE с β=18 и seed=1 имеет большее значение (0.42), чем β-TCVAE с β=4 и seed=4 (0.38).

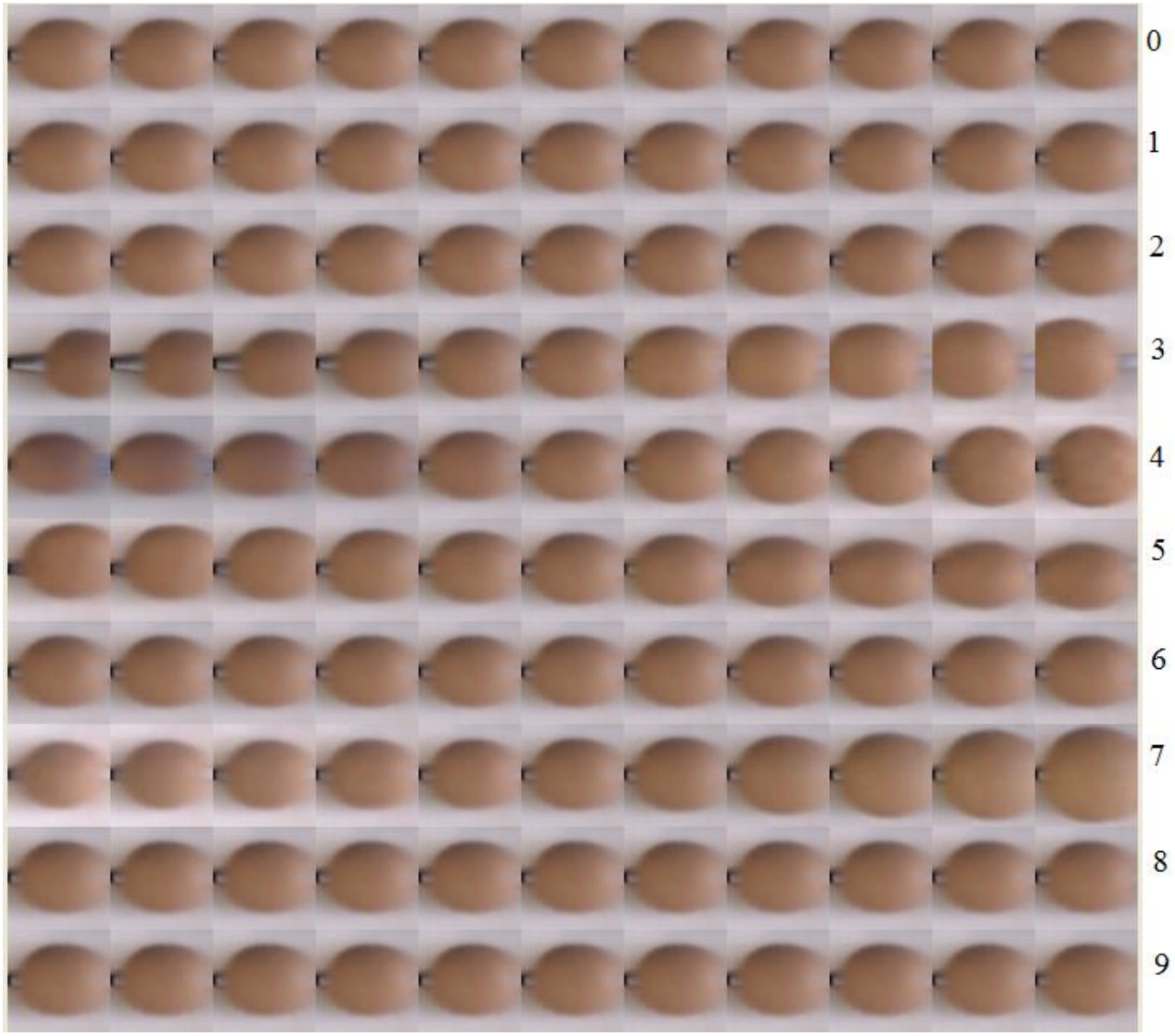

Рисунок 4.21. - Метод варьирования компонент для β-TCVAE с β=18 и seed=1.

4.2.3 Классификация на основе релевантных признаков

Данный этап экспериментов состоит из трех основных частей: определение релевантных признаков, подбор порога для компоненты по небольшой разметке и классификация по компоненте. Как было упомянуто ранее, первую часть можно заменить некоторым алгоритмом машинного обучения. К примеру, можно применить обучение на малом количестве примеров, описанное в пункте 1.4, и ИНС сама отберет релевантные признаки. Или же можно использовать алгоритмы из классического машинного обучения: случайный лес, XGBoost и другие.

Для классификации на основе компонент латентного представления была выбрана модель β-TCVAE (β = 4, seed = 0). Она имеет хорошую распутанность (см. рисунок 4.14), при этом довольно много значимых компонент – 7. Также в сжатом представлении данной модели есть компонента, отвечающая за добавление черных пятен (см. рисунок 4.22, 0-ая компонента).

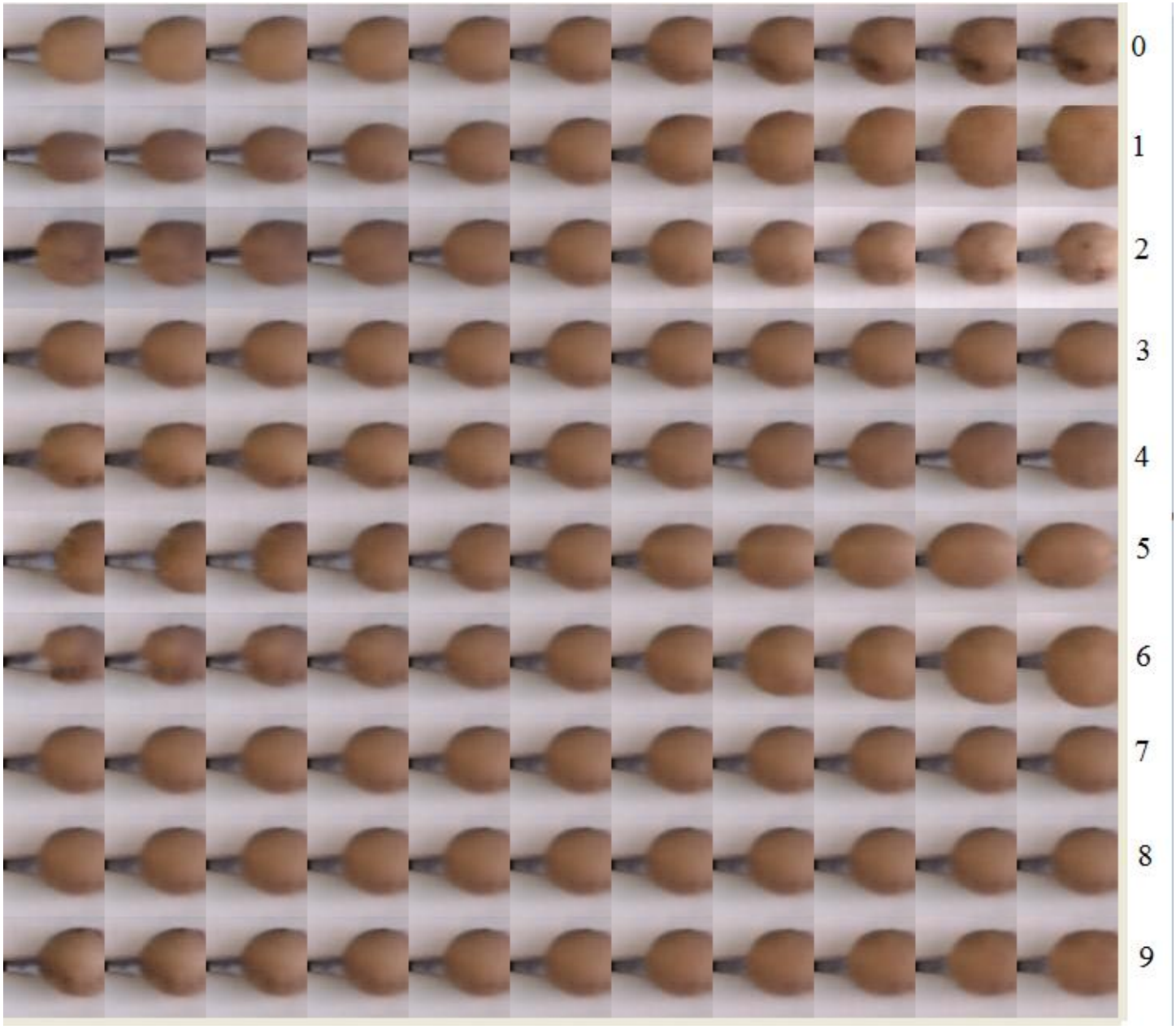

Рисунок 4.22. – 0-ая компонента β-TCVAE (β = 4, seed = 0) отвечает за добавление черных пятен на клубень картофеля.

Именно этот признак и был выбран, как релевантный, для классификации качества картофеля. Для определения порога компоненты использовалась небольшая разметка (100 изображений хорошего и 100 изображений плохого картофеля). На рисунке 4.23 красным точкам соответствуют плохие клубни картофеля, а синим - хорошие. По оси Ox откладывается значение 0-ой компоненты сжатого представления, а по оси Oy – 2-ой. Видно, что данные довольно хорошо разделяются по компоненте. Для классификации был выбран порог 0.1 (если значение компоненты < 0.1, то клубень картофеля классифицировался как хороший, а если ≥ 0.1 – как плохой).

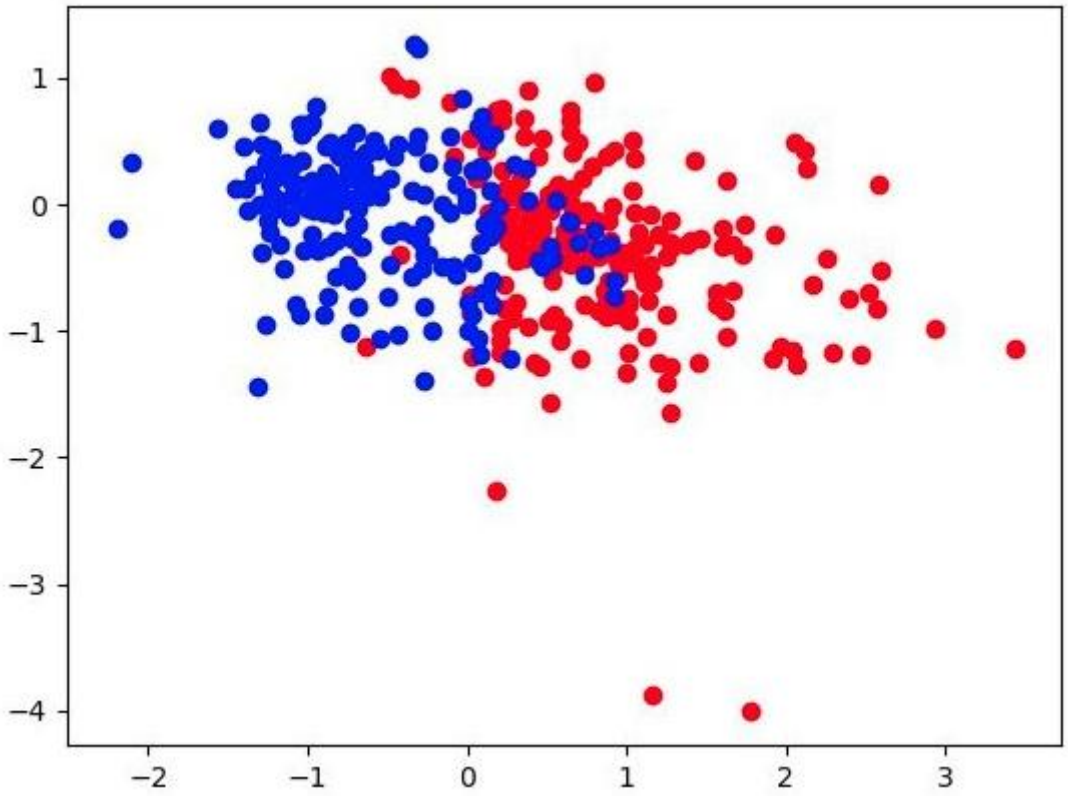

Рисунок 4.23. - красным точкам соответствуют плохие клубни картофеля, а синим - хорошие. По оси Ox откладывается значение 0-ой компоненты β-TCVAE (β = 4, seed = 0), а по оси Oy – 2-ой.

Для данного порога точность классификации на тестовой выборке составила 81.3%.

Среди клубней, которые были проклассифицированы неправильно, довольно много изображений плохого картофеля без черных пятен (рисунок 4.24). Действительно, классификация по 2-ому признаку не учитывает этот случай. Возможно, если бы не было первого этапа отбора релевантных признаков, а сжатые представления сразу подавались в некоторый алгоритм машинного обучения, то данный случай был бы учтен.

Рисунок 4.24. – Примеры неправильно проклассифицированных клубней.

4.3 Архитектуры и параметры обучения

Архитектуры кодировщика и декодера, которые использовались при обучении, описаны в таблице 4.1.

Таблица 4.1. Архитектура кодировщика и декодера, использовавшаяся во всех экспериментах.

Кодировщик	Декодер
Вход: 64 x 64 x 3 1: 4x4 свёртки с шагом 2, 48 фильтров ReLU активация 2: 4x4 свёртки с шагом 2, 48 фильтров ReLU активация 3: 4x4 свёртки с шагом 2, 96 фильтров ReLU активация 4: 4x4 свёртки с шагом 2, 96 фильтров ReLU активация 5: Полносвязный слой, 256 6: Полносвязный слой, 10 для среднего; 6: Полносвязный слой, 10 для логарифма вариации	Вход: $\mathbb{R}^{10}$ 1: Полносвязный, 256 ReLU активация 2: Полносвязный, 4 x 4 x 64 ReLU активация 3: 4x4 обратная свёртка с шагом 2, 96 фильтров ReLU активация 4: 4x4 обратная свёртка с шагом 2, 48 фильтров ReLU активация 5: 4x4 обратная свёртка с шагом 2, 48 фильтров ReLU активация 6: 4x4 обратная свёртка с шагом 2, 3 фильтра

В таблице 4.2 представлены гиперпараметры моделей и в таблице 4.3 представлены общие для всех моделей параметры обучения.

Таблица 4.2 Гиперпараметры моделей

Модель	Параметры	Значения параметров
β-VAE	β	[1, 2, 3, 4, 6, 10, 14, 18]
Cβ-VAE	C_{max} β	[6, 12] (нат) [700, 1400, 2100]
β-TCVAE	β	[2, 3, 4, 6, 10, 14, 18]

В случае с Cβ-VAE в течение обучения C равномерно повышалось от 0 до C_{max}. β подбиралось так, что КЛ-расстояние сходилось достаточно близко к C.

Таблица 4.3. Общие для всех моделей параметры обучения

Параметры	Значения
Количество итераций обучения	300000
Batch size	64
Размерность латентного представления	10
Оптимизатор	Adam
Adam: learning rate	0.0001
Adam: beta1	0.9
Adam: beta2	0.999
Adam: epsilon	0.00000001

4.4 Выводы

При написании практической части использовался язык Python. Python отлично подходит для исследовательских проектов в сфере машинного обучения. Для него написано множество различных полезных бибилиотек и фреймворков: тут и фреймфорки для разработки в сфере глубоко обучения, различные библиотеки для работы с данными, для визуализации и т.д. В то же время Python довольно легко освоить. В качестве основного фреймворка для построения и обучения моделей глубокого обучения был выбран PyTorch. В процессе выполнения данной работы был подробно изучен данный Python-фреймворк и с помощью его инструментов реализованы подготовка, загрузка и предобработка данных. Также с помощью PyTorch также был реализован тренировочный процесс для β-VAE, Cβ-VAE и β-TCVAE.

Для качесвтенной оценки моделей использовался метод варьирования компонент и UDR метрика. Метод варьирования компонент очень нагляден и позволяет понять значения компонент, при этом плохо подходит для сравнения моделей: требует много времени и не даёт однозначного результата, какая модель лучше с точки зрения распутанности, когда это не заметно визуально. Поэтому для предварительного отбора моделей использовалась UDR-метрика. Главное преимущество данной метрики в том, что она не требует разметки по факторам варитивности. Она отлично подходит для подбора гиперпараметров в рамках одной модели, но может страдать от, так называемого, “стадного эффекта” при сравнении различных модификаций VAE. “Стадный эффект” заключается в том, что из-за своей специфики некоторая модификация может выучить отличное от других распутанное представление. Как результат, значение UDR данной модели будет занижено. И наоборот, наибольшее значение UDR может иметь модель, наиболее “похожая” на участвующие в сравнении. Также для отслеживания динамики обучения и оценки особенностей моделей использовались отрисовка распределений компонент латентного представления и качество реконструкции.

В рамках проведенных экспериментов было обучено 115 моделей для получения распутанных представлений. На лучших из них была проведена классификация для определения качества клубней картофеля. После получения распутанных представлений применялся наиболее простой алгоритм: отбирались признаки, релевантные качеству картофеля и по ним проводилась классификация. Например, в случае классификации по представлениям бета-тц-вае, была выбрана компонента, отвечающая за добавления черных пятен на клубень картофеля. Даже такой простой алгоритм дал точность 81.3% на тестовой выборке. При анализе результатов видно, что большинство неверно проклассифицированных клубней принадлежат к классу ‘плохих’, но не имеют черных пятен. Действительно, классификация по признаку, отвечающему за наличие черных пятен, не учитывает этот случай. Использование алгоритмов машинного обучения для автоматического отбора релевантных признаков и классификации могло бы значительно улучшить результат классификации, но данный вопрос не был

исследован в рамках работы, так как основной фокус делался на алгоритмы обучения без учителя и получение распутанных представлений.

ЗАКЛЮЧЕНИЕ

В рамках данной работы подробно рассмотрены современные подходы для получения распутанных представлений и метрики для оценки их качества. На языке Python, с помощью фреймворка PyTorch, разработан инструментарий для обучения моделей, их визуализации с точки зрения распутанности, отслеживания динамики обучения и вычисления метрик распутанности. Были получены распутанные представления для изображений клубней картофеля. В качестве практического применения, на основе полученных представлений, была решена задача классификации качества клубней (точность составила 81.3%). При этом было получено множество других признаков таких, как форма, размер, положение в кадре, которые могут быть полезны для других практических применений. Стоит отметить, что предложенное решение имеет большой потенциал, помимо применения в задаче классификации. Распутанные представления могут быть использованы для обучения на малом количестве примеров (см. пункт 1.4), для нахождения сцен в видеоряде, соответствующих определенным признакам, и для многого-многого другого.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Auto-Encoding Variational Bayes [Электронный ресурс]. –Режим доступа: https://arxiv.org/pdf/1312.6114.pdf
2. β-VAE: learning basic visual concepts with a constrained variational framework [Электронный ресурс]. –Режим доступа: https://openreview.net/references/pdf?id=Sy2fzU9gl.
3. Understanding disentangling in β-VAE [Электронный ресурс]. –Режим доступа: https://arxiv.org/pdf/1804.03599.pdf.
4. Расстояние Кульбака-Лейблера [Электронный ресурс]. –Режим доступа: https://ru.wikipedia.org/wiki/Расстояние_Кульбака_—_Лейблера
5. Документация PyTorch [Электронный ресурс]. –Режим доступа: https://pytorch.org/tutorials/
6. PyTorch – новый фреймворк глубокого обучения [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/334380/
7. Isolating Sources of Disentanglement in Variational Autoencoders [Электронный ресурс]. –Режим доступа: https://arxiv.org/abs/1802.04942
8. Deep Variational Information Bottleneck [Электронный ресурс]. –Режим доступа: https://arxiv.org/abs/1612.00410
9. Deep Learning and Informational Bottleneck Principle [Электронный ресурс]. –Режим доступа: https://arxiv.org/abs/1503.02406
10. The information bottleneck method [Электронный ресурс]. –Режим доступа: https://arxiv.org/abs/physics/0004057
11. Variational Autoencoders Pursue PCA directions (by Accident) [Электронный ресурс]. –Режим доступа: https://arxiv.org/abs/1812.06775
12. Unsupervised Model Selection for Variational Disentangled Representation Learning [Электронный ресурс]. –Режим доступа: https://openreview.net/forum?id=SyxL2TNtvr