**НИТУ «МИСиС»**

Институт ИТКН

**Кафедра инженерной кибернетики**

Направление подготовки:01.03.04 прикладная математика

Квалификация (степень): бакалавр

**КУРСОВАЯ РАБОТА**

**учебная дисциплина**

**«Искусственные нейронные сети»**

Тема: «Автоэнкодер. Структура и особенности применения. Виды автоэнкодеров».

**VII семестр 2020 – 2021 у.г.**

**Учащийся: Репкина Е.А.**

**Группа: БПМ-17-2**

**Проверил: Кондыбаева А.Б.**

**Москва, 2020 г.**

**Содержание**

[**Введение** 3](#_Toc59908578)

[**Основная часть** 4](#_Toc59908579)

[**1 Понятие автоэнкодера** 4](#_Toc59908580)

[**2 Структура автоэнкодера** 4](#_Toc59908581)

[**3 Применение автоэнкодеров** 5](#_Toc59908582)

[**4 Виды автоэнкодеров** 6](#_Toc59908583)

[**4.1 Традиционный автоэнкодер** 6](#_Toc59908584)

[**4.2 Разреженный автоэнкодер** 7](#_Toc59908585)

[**4.3 Вариационный автоэнкодер** 7](#_Toc59908586)

[**4.4 Шумоподавляющий автоэнкодер** 8](#_Toc59908587)

[**5 Реализация модели и результаты** 9](#_Toc59908588)

[**Заключение** 10](#_Toc59908589)

[**Список использованной литературы** 11](#_Toc59908590)

[**ПРИЛОЖЕНИЕ А. Код программы** 12](#_Toc59908591)

# **Введение**

В теории принятия решения очень важное место занимает плотность распределения (или функция распределения) случайных величин. Необходимо иметь оценки функций распределения для расчета апостериорного риска. Оказывается, что автоэнкодеры очень естественны для выполнения оценки функций распределения. Объяснить это можно следующим образом: обучающий набор данных определяется плотностью их распределения. Чем выше плотность обучающих примеров вокруг локальной точки во входном пространстве, тем лучше автоэнкодер реконструирует входной вектор в этом месте пространства.

Сама по себе способность автоэнкодеров сжимать данные используется редко, так как обычно они работают хуже, чем вручную написанные алгоритмы для конкретных типов данных вроде звуков или изображений. А также для них критически важно, чтобы данные принадлежали той генеральной совокупности, на которой сеть обучалась. Обучив автоэнкодер на цифрах, его нельзя применять для кодирования чего-то другого (например, человеческих лиц).

Однако автоэнкодеры можно использовать для предобучения, например, когда стоит задача классификации, а размеченных пар слишком мало. Или для понижения размерности в данных для последующей визуализации. Либо когда просто надо научиться различать полезные свойства входного сигнала.

Более того, некоторые их развития, такие как вариационный автоэнкодер (*VAE*), а также его сочетание с состязающимися генеративным сетями (*GAN*), дают очень интересные результаты и находятся сейчас на переднем крае науки о генеративных моделях.

**Цель данной работы** – изучить структуру и принципы работы автоэнкодера, а также различные виды автоэнкодеров и особенности их применения.

# **Основная часть**

## **1 Понятие автоэнкодера**

**Автоэнкодеры** — это нейронные сети прямого распространения, которые восстанавливают входной сигнал на выходе. Задача автоэнкодера – изучить представление (кодирование) набора данных, обычно для уменьшения размерности, путем обучения сети игнорированию «шума» сигнала. Наряду со стороной сокращения изучается сторона восстановления, где автоэнкодер пытается сгенерировать из сокращенного кодирования представление, максимально приближенное к исходному входу (Рисунок 1), отсюда и его название.

Существует несколько вариантов базовой модели с целью заставить усвоенные представления входных данных принять полезные свойства. Обычно их ограничивают в размерности *кода* (он меньше, чем размерность сигнала) или штрафуют за активации в *коде*. Входной сигнал восстанавливается с ошибками из-за потерь при кодировании, но, чтобы их минимизировать, сеть вынуждена учиться отбирать наиболее важные признаки.

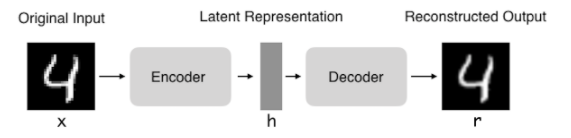


Рисунок 1 – Принцип работы автоэнкодера

## **2 Структура автоэнкодера**

Автоэнкодер состоит из двух частей (Рисунок 2):

1. Энкодер : отвечает за сжатие входа в latent-space. Представлен функцией кодирования ;
2. Декодер : предназначен для восстановления ввода из latent-space. Представлен функцией декодирования .

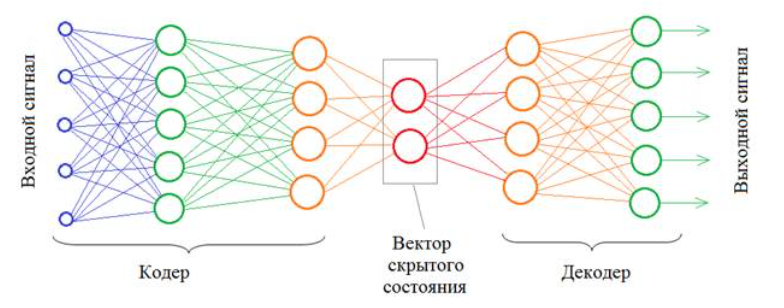


Рисунок 2 – Схема автоэнкодера

Автоэнкодер, изменяя *f* и , стремится выучить тождественную функцию , минимизируя какой-то функционал ошибки.

Простейший автокодировщик состоит из двух полносвязных слоёв, которые описываются преобразованиями следующего вида:

где функция – функция активации (сигмоидальная функция, гиперболический тангенс, ReLU).

Обучение сводится к решению задачи минимизации функционала по набору параметров :

В качестве функции ошибки выбирается квадратичная функция, либо кросс-энтропия.

## **3 Применение автоэнкодеров**

Два основных практических применения автоэнкодеров для визуализации данных:

* сглаживание шума;
* снижение размерности.

С соответствующими ограничениями по размерности и разреженности автоэнкодеры могут изучать data projections, которые более интересны, чем PCA (метод главных компонент) или другие базовые техники.

Автоэнкодеры обучаются автоматически на примерах данных. Это означает, что легко натренировать части алгоритма, которые будут затем хорошо работать на конкретном типе ввода и не будут требовать применения новой техники, а только соответствующие данные для обучения.

Однако автоэнкодеры будут плохо справляться со сжатием изображений. По мере того, как автокодер обучается по заданному датасету, он достигает разумных результатов сжатия, аналогичных используемому для тренировок набору, но плохо работает как компрессор общего назначения. Сжатие JPEG, например, будет справляться намного лучше.

## **4 Виды автоэнкодеров**

Основые виды автокодировщиков:

* Традиционные автокодировщики
* Разреженные автокодировщики
* Вариационный автокодировщик
* Шумоподавляющий автокодировщик

Рассмотрим подробнее каждый из типов.

### **4.1 Традиционный автоэнкодер**

**Автокодировщик** (autoencoder, AE). Нейросеть по виду напоминает песочные часы, т. к. в ней скрытые слои меньше выходного и входного, при этом она симметрична. Для обучения можно использовать обратное распространение ошибки, подавая входную информацию и задавая ошибку, равную разнице между выходом и входом.

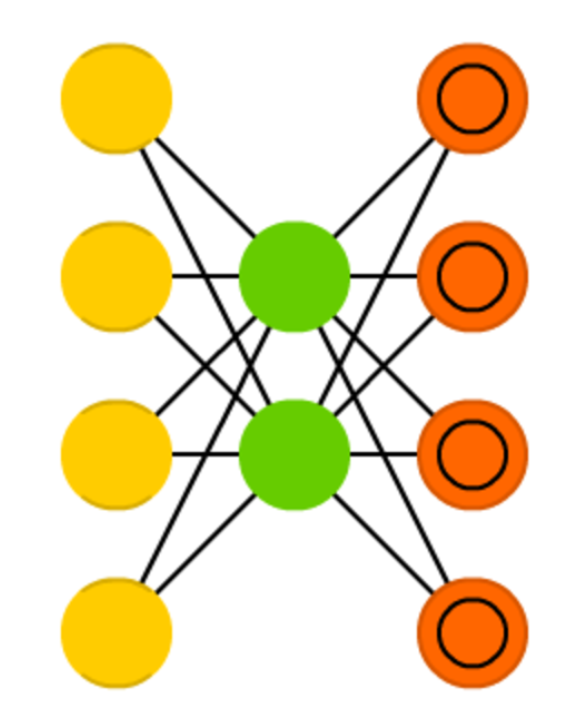


Рисунок 3 – Схема обычного автоэнкодера

### **4.2 Разреженный автоэнкодер**

**Разреженный автокодировщик***(sparse autoencoder, SAE)* — в каком-то смысле противоположность обычного. Вместо того, чтобы обучать сеть отображать информацию в меньшем «объёме» узлов, мы увеличиваем их количество. Вместо того, чтобы сужаться к центру, сеть там раздувается. Сети такого типа полезны для работы с большим количеством мелких свойств набора данных. Если обучать сеть как обычный автокодировщик, ничего полезного не выйдет. Поэтому кроме входных данных подаётся ещё и специальный фильтр разреженности, который пропускает только определённые ошибки.

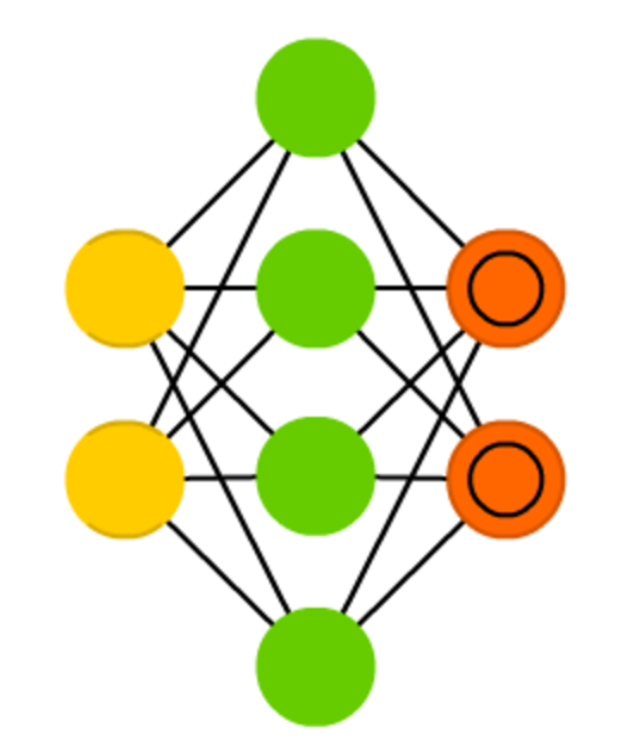


Рисунок 4 – Схема разреженного автоэнкодера

### **4.3 Вариационный автоэнкодер**

**Вариационные автокодировщики**(variational autoencoder, VAE) обладают схожей с AE архитектурой, но обучают их иному: приближению вероятностного распределения входных образцов. В этом они берут начало от машин Больцмана. Тем не менее, они опираются на байесовскую математику, когда речь идёт о вероятностных выводах и независимости, которые интуитивно понятны, но сложны в реализации. Если обобщить, то можно сказать что эта сеть принимает в расчёт влияния нейронов. Если что-то одно происходит в одном месте, а что-то другое — в другом, то эти события не обязательно связаны, и это должно учитываться.

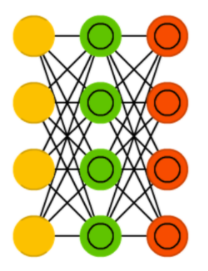


Рисунок 5 – Схема вариационного автоэнкодера

### **4.4 Шумоподавляющий автоэнкодер**

**Шумоподавляющие автокодировщики** (denoising autoencoder, DAE) — это AE, в которые входные данные подаются в зашумленном состоянии. Ошибку мы вычисляем так же, и выходные данные сравниваются с зашумленными. Благодаря этому сеть учится обращать внимание на более широкие свойства, поскольку маленькие могут изменяться вместе с шумом.

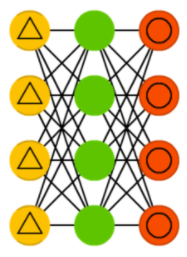


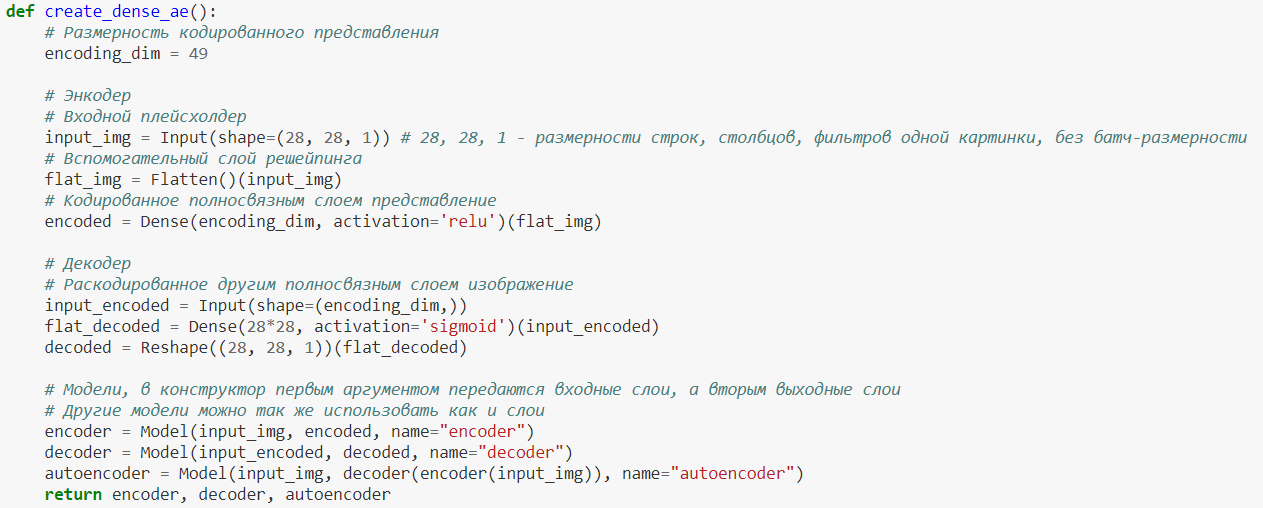
Рисунок 6 – Схема шумоподавляющего автоэнкодера

## **5 Реализация модели и результаты**

Для реализации модели используется библиотека глубоко обучения Keras. В качестве исходных данных возьмём базу данных образцов рукописного написания цифр MNIST.

Создадим простой (сжимающий, undercomplete) автоэнкодер скодом малой размерности из двух полносвязных слоев: енкодера и декодера.

Создание модели автоэнкодера:



Обучение производилось на 50 эпохах.

На Рисунке 7 приведены оригинальные изображения и изображения, полученные в результате работы модели. Как видно из рисунка, цифры воспроизведены достаточно точно, при этом они стали более размытыми и сглаженными.



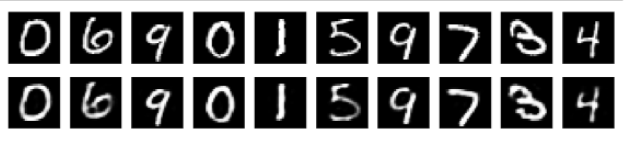


Рисунок 7 – Сравнение оригиналов с декодированными изображениями цифр

# **Заключение**

Автокодировщик – это впечатляющая техника машинного обучения без учителя, во многих задачах позволившая превзойти методы ручного отбора признаков. В ходе курсовой работы была изучена структура автоэнкодера, особенности его применения, рассмотрены различные виды автоэнкодеров и реализована модель автоэнкодера.

# **Список использованной литературы**

1. <https://www.deeplearningbook.org/contents/autoencoders.html>
2. <https://habr.com/en/post/331382/>
3. <https://www.machinelearningmastery.ru/a-high-level-guide-to-autoencoders-b103ccd45924/>
4. <https://proproprogs.ru/neural_network/avtoenkodery-chto-eto-kak-rabotayut>
5. <https://tproger.ru/translations/neural-network-zoo-1/>

# **ПРИЛОЖЕНИЕ А. Код программы**

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.datasets import mnist # библиотека базы выборок Mnist

from tensorflow import keras

from tensorflow.keras.layers import Dense, Flatten, Reshape, Input

from keras.models import Model

# Загрузка датасета рукописных цифр MNIST

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

# Стандартизация входных данных

x\_train = x\_train.astype('float32') / 255.

x\_test = x\_test .astype('float32') / 255.

x\_train = np.reshape(x\_train, (len(x\_train), 28, 28, 1))

x\_test = np.reshape(x\_test, (len(x\_test), 28, 28, 1))

# Модель автоэнкодера

def create\_dense\_ae():

# Размерность кодированного представления

encoding\_dim = 49

# Энкодер

# Входной плейсхолдер

input\_img = Input(shape=(28, 28, 1)) # 28, 28, 1 - размерности строк, столбцов, фильтров одной картинки, без батч-размерности

# Вспомогательный слой решейпинга

flat\_img = Flatten()(input\_img)

# Кодированное полносвязным слоем представление

encoded = Dense(encoding\_dim, activation='relu')(flat\_img)

# Декодер

# Раскодированное другим полносвязным слоем изображение

input\_encoded = Input(shape=(encoding\_dim,))

flat\_decoded = Dense(28\*28, activation='sigmoid')(input\_encoded)

decoded = Reshape((28, 28, 1))(flat\_decoded)

# Модели, в конструктор первым аргументом передаются входные слои, а вторым выходные слои

# Другие модели можно так же использовать как и слои

encoder = Model(input\_img, encoded, name="encoder")

decoder = Model(input\_encoded, decoded, name="decoder")

autoencoder = Model(input\_img, decoder(encoder(input\_img)), name="autoencoder")

return encoder, decoder, autoencoder

# Создание и компиляция модели

encoder, decoder, autoencoder = create\_dense\_ae()

autoencoder.compile(optimizer='adam', loss='binary\_crossentropy')

# Число параметров

autoencoder.summary()

# Обучаем автоэнкодер

autoencoder.fit(x\_train, x\_train,

epochs=50,

batch\_size=256,

shuffle=True,

validation\_data=(x\_test, x\_test))

# Функция отрисовки цифр

%matplotlib inline

import seaborn as sns

def plot\_digits(\*args):

args = [x.squeeze() for x in args]

n = min([x.shape[0] for x in args])

plt.figure(figsize=(2\*n, 2\*len(args)))

for j in range(n):

for i in range(len(args)):

ax = plt.subplot(len(args), n, i\*n + j + 1)

plt.imshow(args[i][j])

plt.gray()

ax.get\_xaxis().set\_visible(False)

ax.get\_yaxis().set\_visible(False)

plt.show()

# Закодируем изображения

n = 20

imgs = x\_test[:n]

encoded\_imgs = encoder.predict(imgs, batch\_size=n)

encoded\_imgs[0]

# Декодируем изображения

decoded\_imgs = decoder.predict(encoded\_imgs, batch\_size=n)

# Сравнение результата с оригиналом

plot\_digits(imgs, decoded\_imgs)