УНИВЕРСИТЕТ
искусственного
интеллекта

Автокодировщики

Автокодировщики

Автокодировщик (англ. autoencoder) – специальная архитектура искусственных нейронных сетей, позволяющая применять обучение без учителя при использовании метода обратного распространения ошибки.

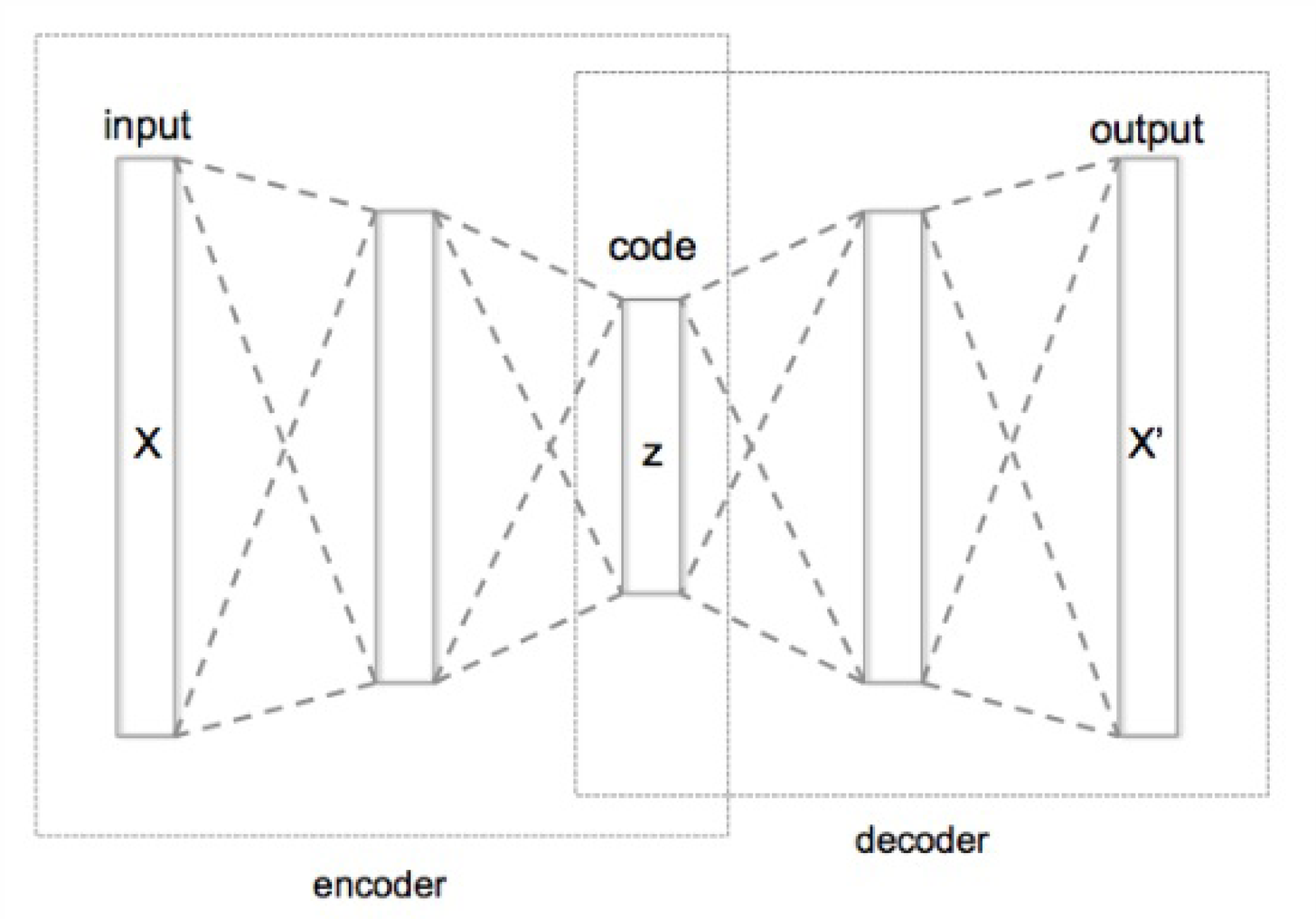
Цель обучения автокодировщиков: найти внутреннюю структуру в данных.

Применение автокодировщиков:

- улучшение качества изображения (увеличение размера, шумоподавление и т. д.),
- генерация новых данных по заданному образцу (например, раскрашивание черно-белых картинок в цветные),
- поиск выбросов.

Простейшая архитектура автокодировщика – сеть прямого распространения без обратных связей, наиболее схожая с персептроном и содержащая входной слой, промежуточный слой и выходной слой. В отличие от персептрона выходной слой автокодировщика должен содержать столько же нейронов, сколько и входной слой.

Автокодировщики

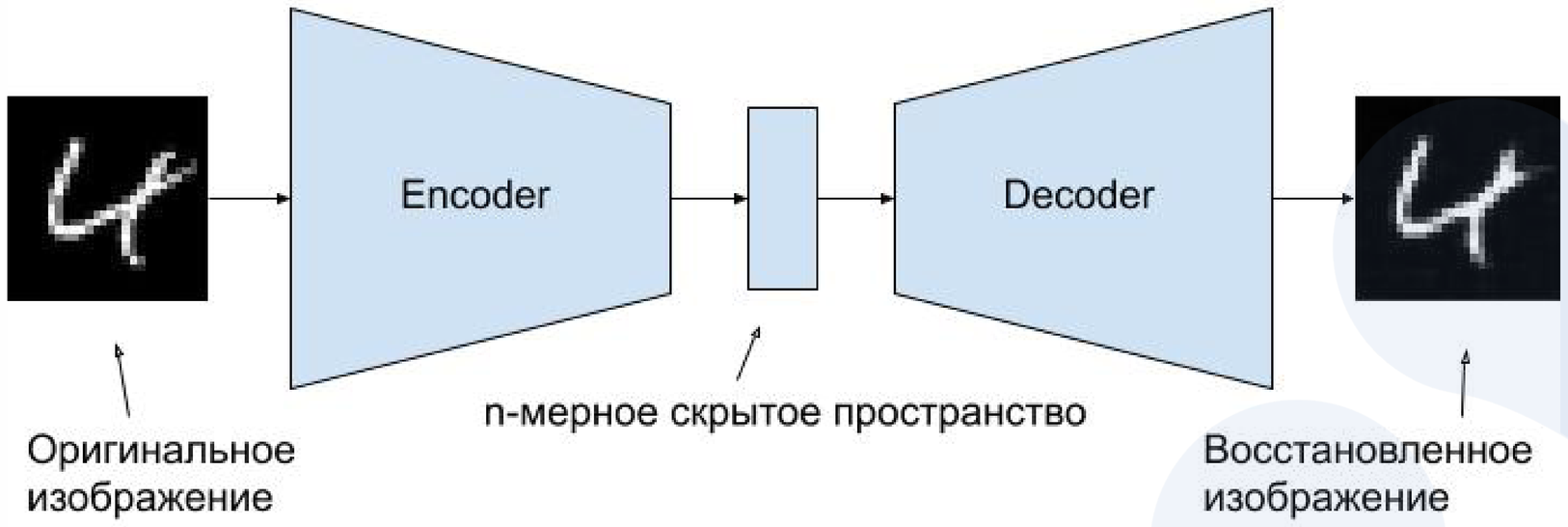


Автокодировщик состоит из двух частей: **энкодера** (кодирует выборку X в свое внутреннее представление Z) X и **декодера** (восстанавливает исходную выборку) X'.

Таким образом, автокодировщик пытается совместить восстановленную версию каждого объекта выборки с исходным объектом.

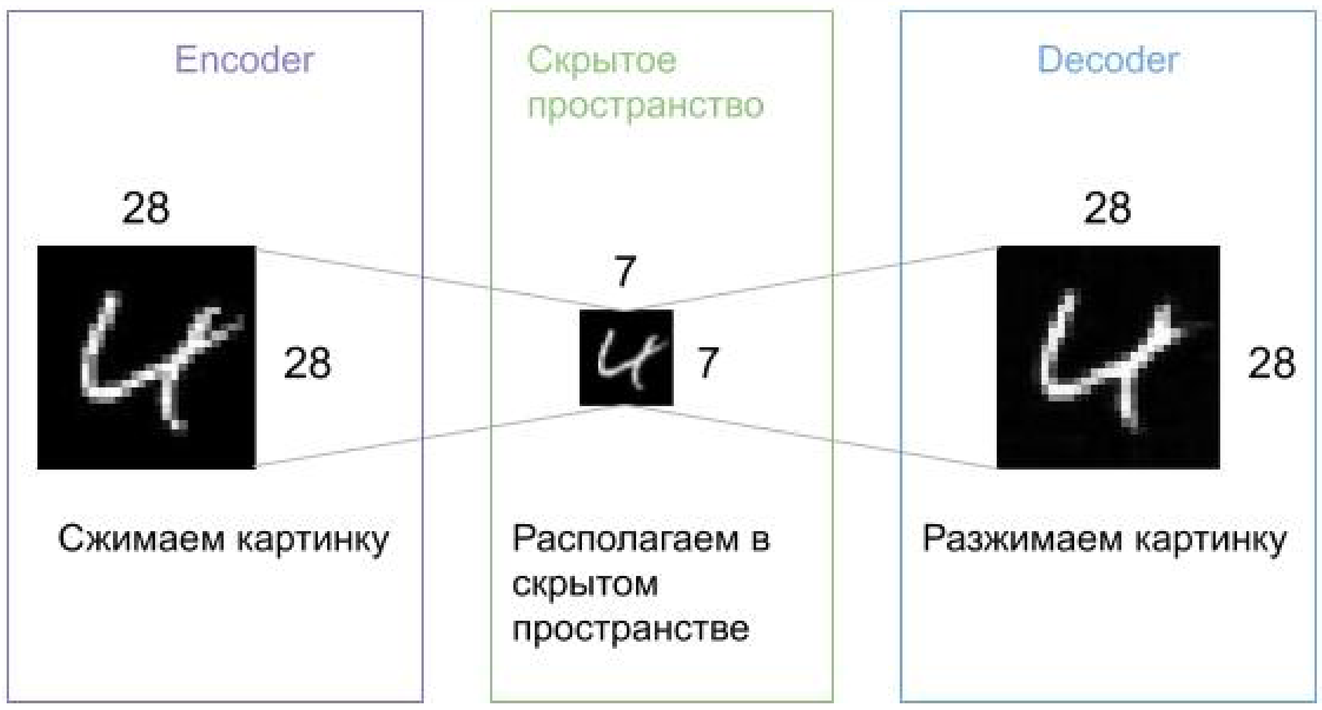
Рассмотрим пример работы автоэнкодера на восстановлении изображений из библиотеки mnist.

На вход сеть принимает объект X (цифра 4).



Автокодировщики

Наша сеть кодирует этот объект в скрытое состояние. Затем по скрытому состоянию восстанавливается реконструкция объекта X', которая должна быть похожа на X. Как мы видим, восстановленное изображение (справа) стало более размытым. Объясняется это тем, что мы стараемся сохранить в скрытом представлении только наиболее важные признаки объекта, поэтому объект восстанавливается с потерями.



Одно из основных предназначений таких автокодировщиков – снижение размерности исходного пространства. Когда мы имеем дело с автокодировщиками, то сама процедура тренировки нейросети заставляет автокодировщик запоминать основные признаки объектов, по которым будет проще восстановить исходные объекты выборки.

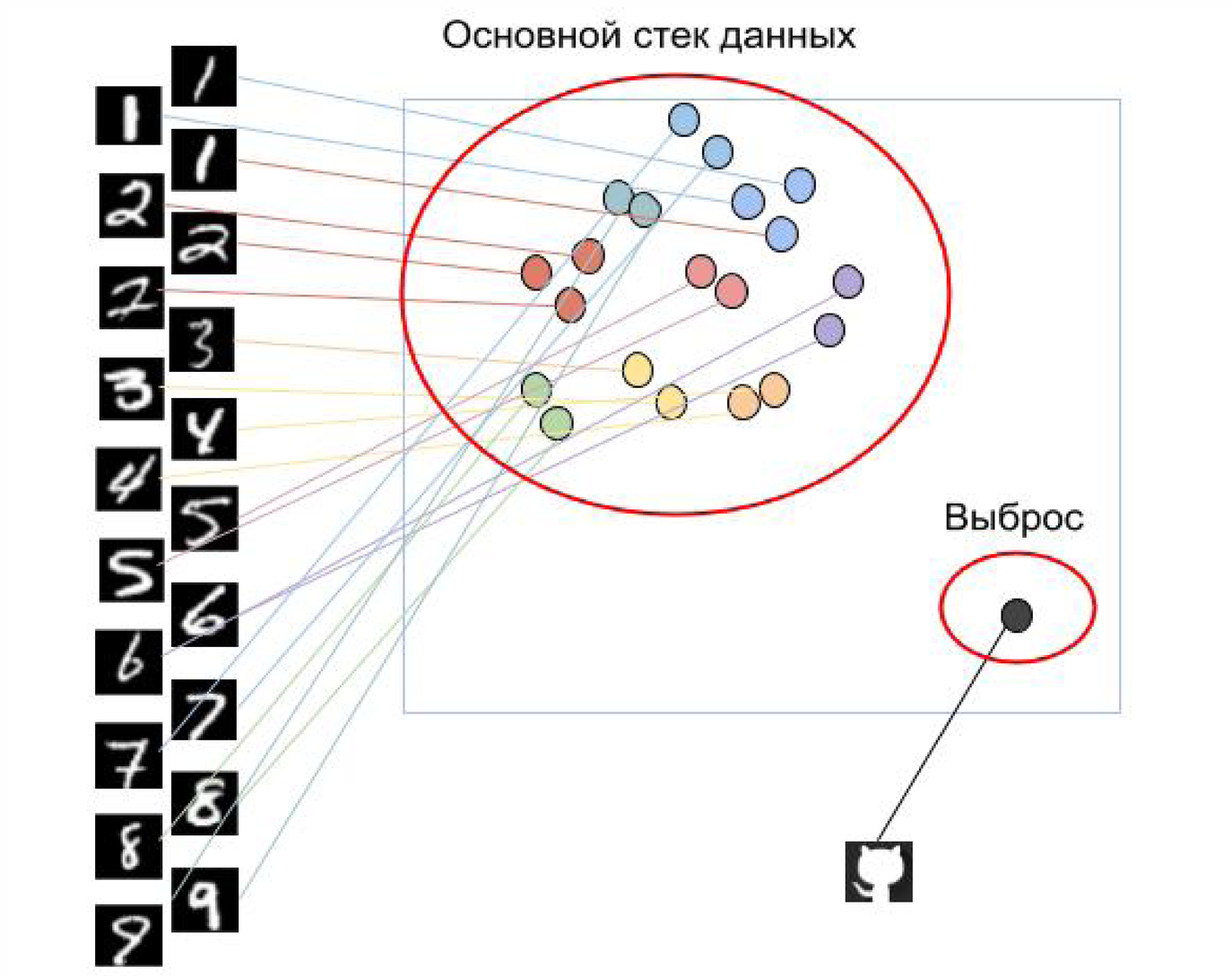
Мы можем настроить архитектуру сети таким образом, что она будет увеличивать размер изображений, например, из 100x100 пикселей в 200x200 без потери качества.

Также сеть неплохо справляется с задачей подавления шума на фотографиях.

Поиск выбросов

Выбросы – объекты, которые встречаются очень редко (люди в возрасте больше 100 лет, мошеннические транзакции, успешные музыкальные группы и т. п.).

Автокодировщики



Например, мы обучаем сеть на датасете рукописных цифр от 1 до 9, в скрытом пространстве они будут находиться в области неподалеку друг от друга, сформировавшись в некоторые группы (классы), соответствующие своему значению.

Если подать на вход такой сети какую-то картинку (силуэт кота), отличающуюся от рукописных цифр, то она будет располагаться в стороне от основной массы данных. Это и есть выброс (объект, который не относится к той базе, на которой обучался автокодировщик).

Где это можно использовать?

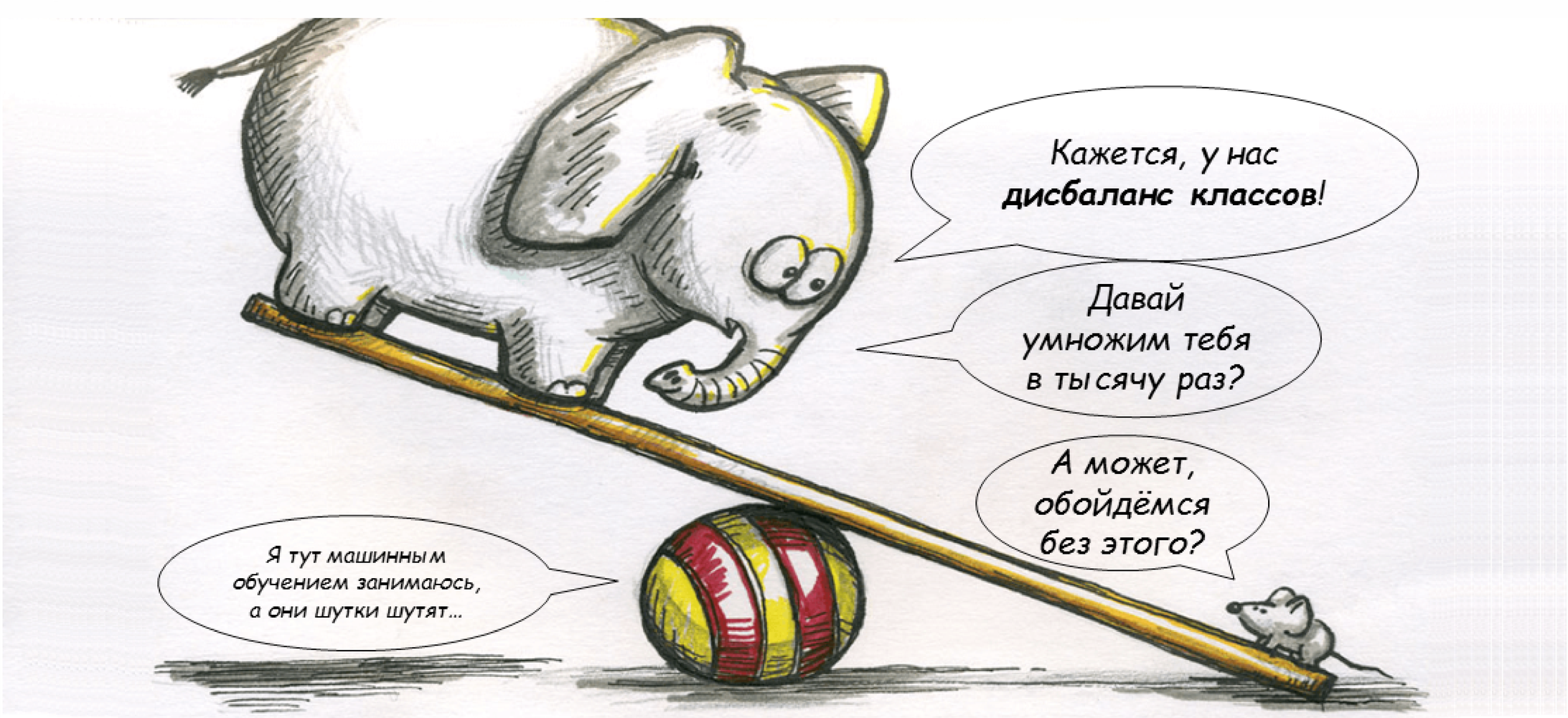
Там, где имеется сильная разбалансировка классов. Например, работа атомной электростанции. Мы хотим обучить сеть распознавать аварийные ситуации на основе данных, приходящих с датчиков станции. Если бы мы использовали метод обучения с учителем, нам понадобилась бы база с нормальной работой станции и аварийными случаями (количество аварийных случаев должно быть примерно равно количеству данных нормальной работы).

Но так как мы имеем дело с атомной станцией, количество аварийных ситуаций (а значит и данных при аварийной ситуации) очень мало, и

Автокодировщики

поэтому мы не сможем обучить сеть способом классического обучения с учителем.

Вот здесь и помогают автокодировщики. Так как у нас есть база нормальной (безаварийной) работы станции, то любое *отклонение*



от нормы будет *выбросом* автокодировщика и сигналом к аварийной ситуации.

Далее рассмотрим пример построения модели автокодировщиков для поиска выбросов на базе мошеннических операций с банковскими картами. База имеет 282000 нормальных операций и 492 мошеннических, значит мы будем обучать сеть на нормальных данных и потом спредиктим на мошеннических, для того чтобы распознать выброс. Значение выброса мы зададим сами. Датасет состоит из 30 столбцов, значения которых мы будем подавать сети.

Модель построения автокодировщика для поиска выбросов

- Обучаем автокодировщик только на обычных данных

```
dataSize = X_train.shape[1] # берём размеры X_
train(количество столбцов обучающей базы)
dataInput = Input(shape=(dataSize, )) # задаем эти
размеры как входные в сеть
x = Dense(10, activation='relu')(dataInput) # пропускаем
```

Автокодировщики

```
через полносвязный слой размером 10
x = Dense(dataSize, activation='linear')(x) # и через
полносвязный слой размером 30
autoencoder = Model(inputs=dataInput, outputs=x) #
собрали модель
autoencoder.compile(optimizer='Adam', loss='mse')
history = autoencoder.fit(X_train, X_train,
                          epochs=5,
                          batch_size=32)
```

В качестве **и** x_train, **и** y_train подаем значения X_train.

- При распознавании оцениваем, насколько хорошо автокодировщик восстановил данные на выходе

```
predictions = model.predict(X_test) # делаем предсказание
по X_test
mse = np.mean(np.power(X_test - predictions, 2),
axis=1) # определяем среднеквадратичную ошибку по X_
test и предсказанием по X_test. X_test это данные с
мошенническими операциями.
mse_normal = mse[y_test.values == 0] # среднеквадратичная
ошибка на нормальных операциях
mse_frauds = mse[y_test.values == 1] # среднеквадратичная
ошибка на мошеннических операциях
print("Минимальная ошибка нормальных транзакций:",
round(min(mse_normal),4)) #найдем минимальную
среднеквадратичную ошибку на нормальных операциях
print("Максимальная ошибка нормальных транзакций:",
round(max(mse_normal),4)) #найдем максимальную
среднеквадратичную ошибку на нормальных операциях
print("Средняя ошибка нормальных транзакций:",
round(sum(mse_normal) / len(mse_normal),4)) # среднюю
ошибку
```

Автокодировщики

```
Минимальная ошибка нормальных транзакций: 0.0461
Максимальная ошибка нормальных транзакций: 354.6082
Средняя ошибка нормальных транзакций: 0.3558
```

- Автокодировщик не обучался на выбросах, поэтому будет плохо их восстанавливать

```
print("Минимальная ошибка мошеннических транзакций:",
round(min(mse_frauds),4)) #найдем минимальную
среднеквадратичную ошибку на мошеннических операциях
print("Максимальная ошибка мошеннических транзакций:",
round(max(mse_frauds),4)) #найдем максимальную
среднеквадратичную ошибку на мошеннических операциях
print("Средняя ошибка мошеннических транзакций:",
round(sum(mse_frauds) / len(mse_frauds),4)) # среднюю
ошибку
```

```
Минимальная ошибка мошеннических транзакций: 0.1621
Максимальная ошибка мошеннических транзакций: 113.4951
Средняя ошибка мошеннических транзакций: 19.8923
```

- Ошибка восстановления больше заданного порога, значит данный объект является выбросом

```
def getAccByBias(bias): # функция будет принимать какое-
то пороговое значение
  isNormal = mse_normal < bias # если ошибка меньше
порога, то транзакция нормальная
  isFrauds = mse_frauds > bias # если ошибка больше
порога, то транзакция мошенническая

  accNormal = sum(isNormal) / len(isNormal) # вычисляем
процент нормальных операций
  accFaruds = sum(isFrauds) / len(isFrauds) # вычисляем
процент мошеннических операций

  print("Распознано нормальных транзакций: ",
round(100*accNormal), "%", sep="")
```

Автокодировщики

```
  print("Распознано мошеннических транзакций: ",
round(100*accFaruds), "%", sep="")
  print("Средняя точность распознавания: ",
round(50*(accNormal + accFaruds)), "%", sep="")
```

Выведем данные по распознаванию с конкретным пороговым значением

```
getAccByBias(1.5)

_____________________
Распознано нормальных транзакций: 99.0%
Распознано мошеннических транзакций: 85.0%
Средняя точность распознавания: 92.0%
```