онлайн-курс

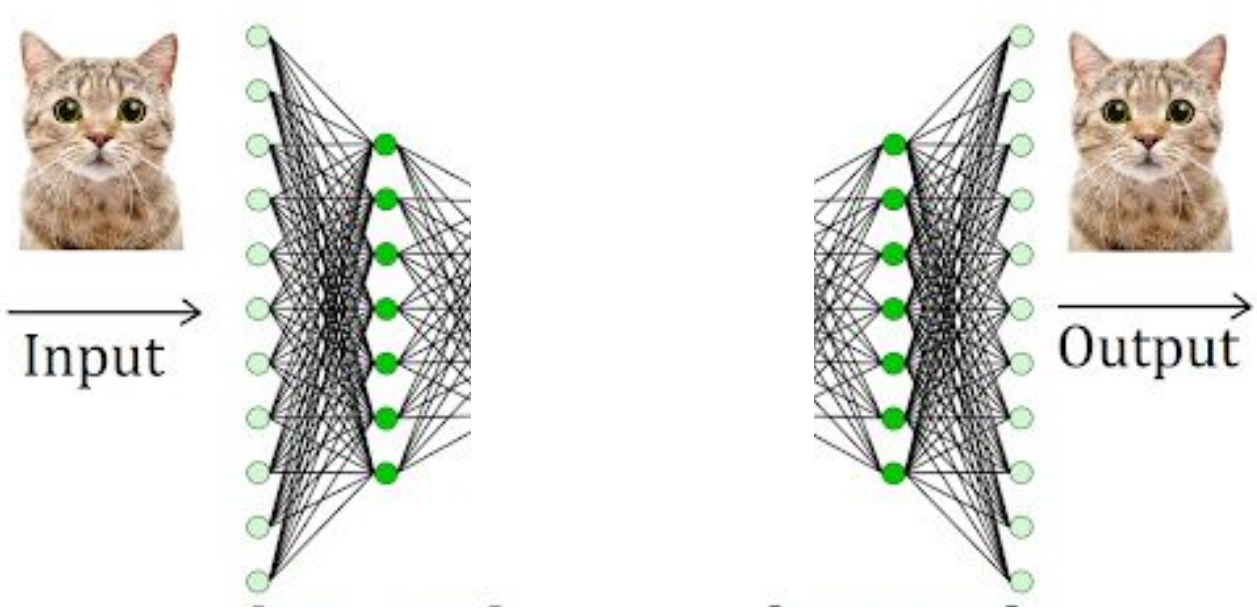
СПЕЦИАЛЬНЫЕ АРХИТЕКТУРЫ НЕЙРОННЫХ СЕТЕЙ

Автокодировщики (АК)

Сделать из конфетки… конфетку

Постановка задачи

Давайте замутим НС, которая должна выдать ответ, в точности равный входу (вход и выход не обязательно могут быть картинками).



А зачем это надо?

Неужели так сложно подобрать веса НС, чтобы она осуществляла бы тождественное преобразование. То есть ее функция равна $F_{NN}(x)=x$.

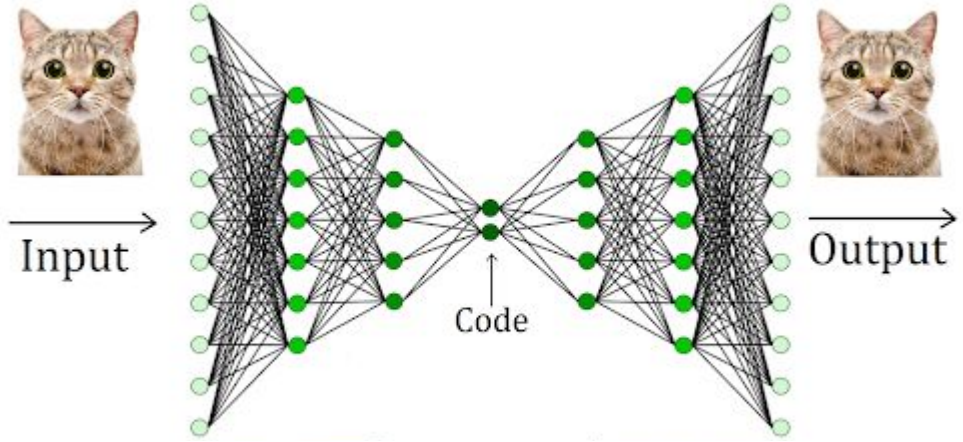
Конечно, сделать так, чтобы внутренние нейроны перенесли всю информацию от начала к концу НС, не сложно.

А мы возьмём и усложним жизнь нашей НС!

Бутылочное горлышко

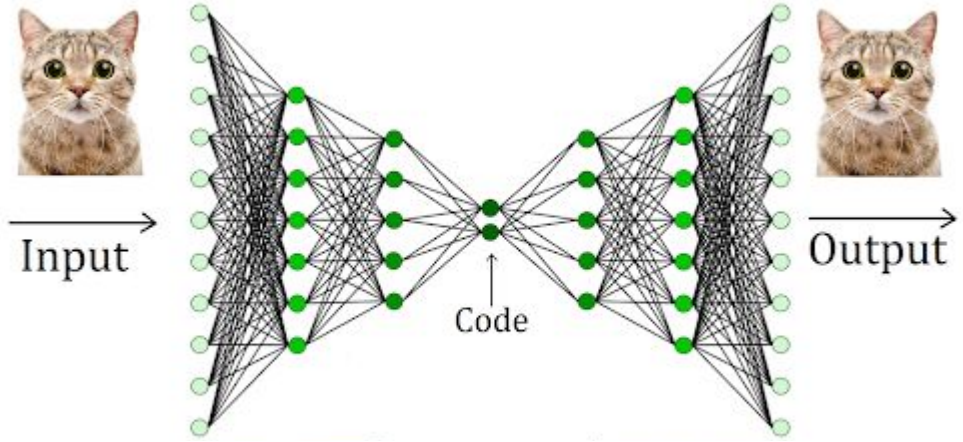
А давайте сделаем специально слой в центре НС очень узким, то есть число нейронов на этом слое меньше размерности входа (выхода) НС.

Что это даст?

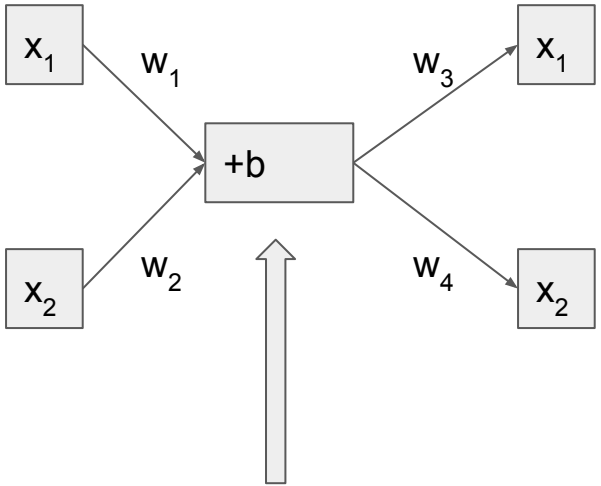


Бутылочное горлышко — смысл

Мы **заставляем** НС находить в объекте самую важную информацию и только ее передавать через бутылочное горлышко.



Пример: данные с зависимостями

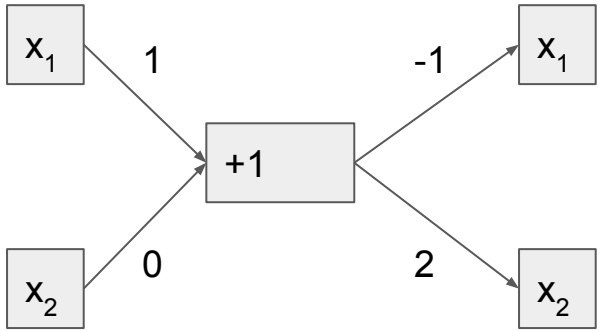


бутылочное горлышко

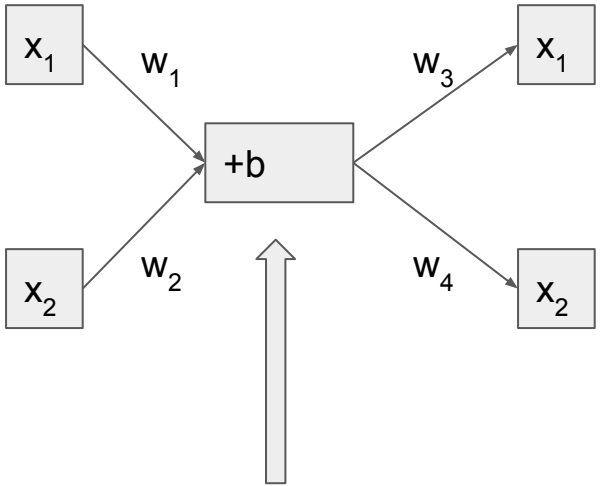
тренировочная выборка

x_1	x_2
1	4
2	6
3	8
4	10

На данной ТВ НС натренируется следующим образом



Пример: сложная зависимость



бутылочное горлышко

тренировочная выборка

x_1	x_2
0	0
1	1
2	4
3	9

Как подобрать оптимальные веса автокодировщика?
Для этого нужно составить **функцию потерь автокодировщика**!

Функция потерь автокодировщика

Если на вход АК подается вектор чисел $x=(x_1,...,x_n)$, то функция сети $F_{NN}(x)$ — это тоже вектор $F_{NN}(x)=(F_1(x),...,F_n(x))$.

То есть
число x_1 превращается в число $F_1(x)$,

число x_2 превращается в число $F_2(x)$,
...
число x_n превращается в число $F_n(x)$.

Составим квадрат разности чисел x_i и того, во что они превратились. Просуммируем:

$L(x)=(F_1(x)-x_1)^2+(F_2(x)-x_2)^2+...+(F_n(x)-x_n)^2$ — **потери на кодировке объекта x**

Функция потерь автокодировщика

А чтобы получить функцию потерь для всей ТВ, необходимо **просуммировать потери всех объектов ТВ**. В нашем примере имеем:

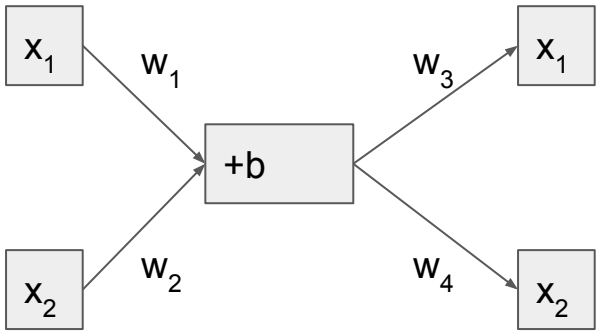
Функция сети для входа $x=(x_1,x_2)$ равна $F_{NN}(x)=(F_1(x),F_2(x))$, где

$F_1(x)=(w_1x_1+w_2x_2+b)w_3$ $\quad$ $F_2(x)=(w_1x_1+w_2x_2+b)w_4$

Потери на первом объекте равны:

$((w_10+w_20+b)w_3-0)^2+((w_10+w_20+b)w_4-0)^2=(bw_3)^2+(bw_4)^2$

Потери всех объектов из ТВ занесём в таблицу.



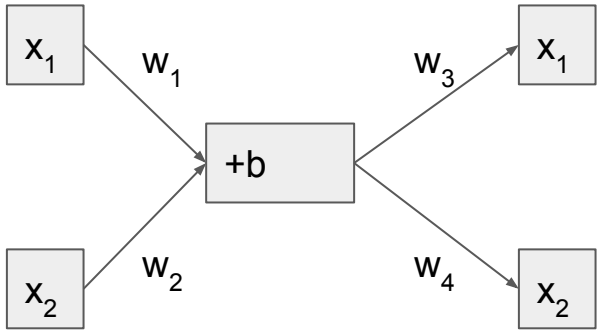
x_1	x_2
0	0
1	1
2	4
3	9

Функция потерь автокодировщика

x_1	x_2	Потери
0	0	$((w_1*0+w_2*0+b)w_3-0)^2+((w_1*0+w_2*0+b)w_4-0)^2$
1	1	$((w_1*1+w_2*1+b)w_3-1)^2+((w_1*1+w_2*1+b)w_4-1)^2$
2	4	$((w_1*2+w_2*4+b)w_3-2)^2+((w_1*2+w_2*4+b)w_4-4)^2$
3	9	$((w_1*3+w_2*9+b)w_3-3)^2+((w_1*3+w_2*9+b)w_4-9)^2$

Итоговая функция потерь равна сумме:

$$L(w)=((w_1*0+w_2*0+b)w_3-0)^2+((w_1*0+w_2*0+b)w_4-0)^2+$$
$$+((w_1*1+w_2*1+b)w_3-1)^2+((w_1*1+w_2*1+b)w_4-1)^2+$$
$$+((w_1*2+w_2*4+b)w_3-2)^2+((w_1*2+w_2*4+b)w_4-4)^2+$$
$$+((w_1*3+w_2*9+b)w_3-0)^2+((w_1*3+w_2*9+b)w_4-9)^2$$



Функция потерь автокодировщика

Итоговая функция потерь равна сумме:

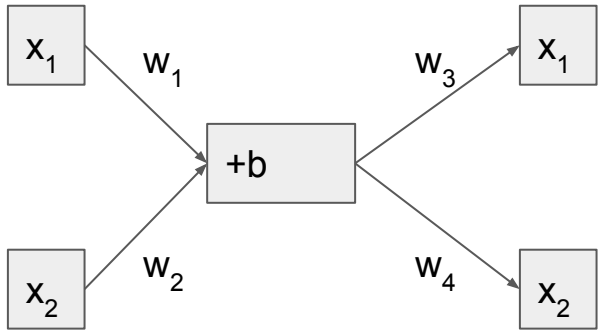
$$L(w)=((w_1*0+w_2*0+b)w_3-0)^2+((w_1*0+w_2*0+b)w_4-0)^2+$$

$$+((w_1*1+w_2*1+b)w_3-1)^2+((w_1*1+w_2*1+b)w_4-1)^2+$$

$$+((w_1*2+w_2*4+b)w_3-2)^2+((w_1*2+w_2*4+b)w_4-4)^2+$$

$$+((w_1*3+w_2*9+b)w_3-0)^2+((w_1*3+w_2*9+b)w_4-9)^2$$

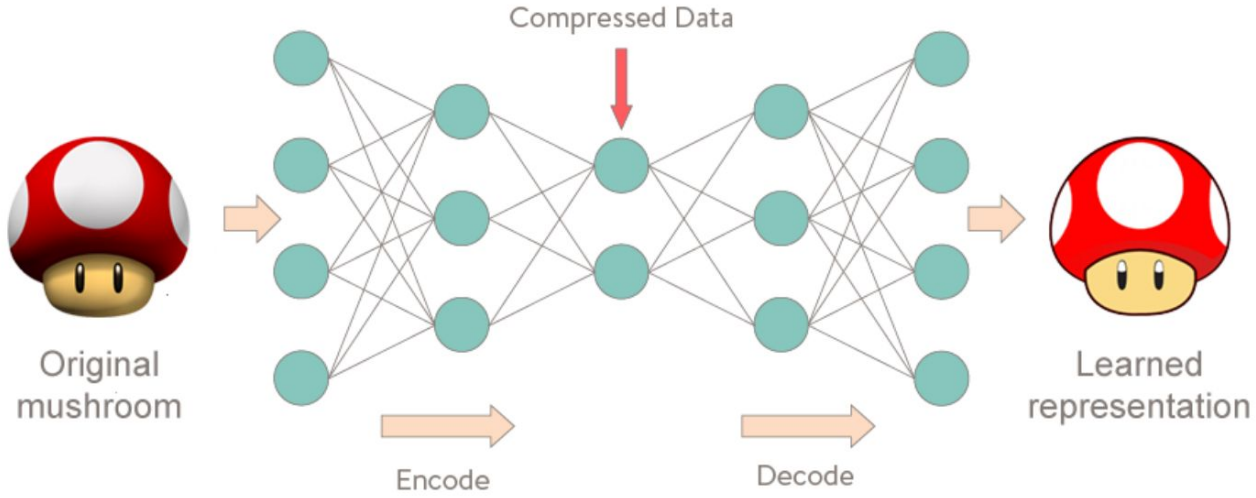
Осталось найти минимум
этой функции с помощью ГС.



Зачем они нужны?

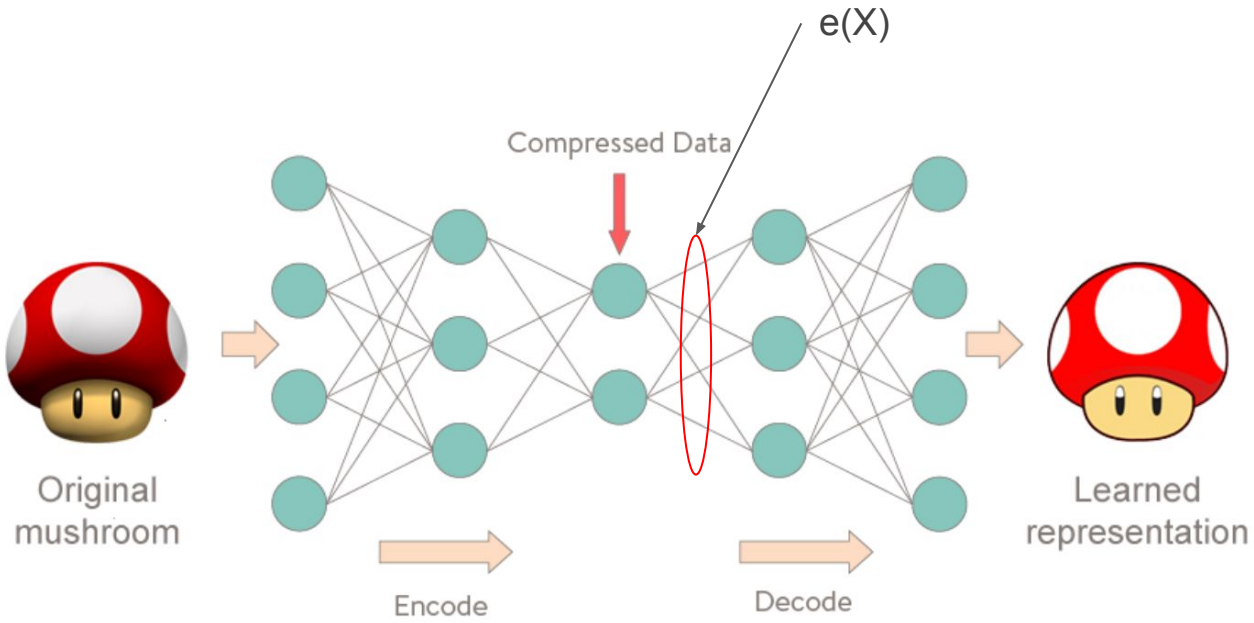
Название частей АК

- кодирующая часть
- бутылочное горлышко (слой сжатия)
- декодирующая часть



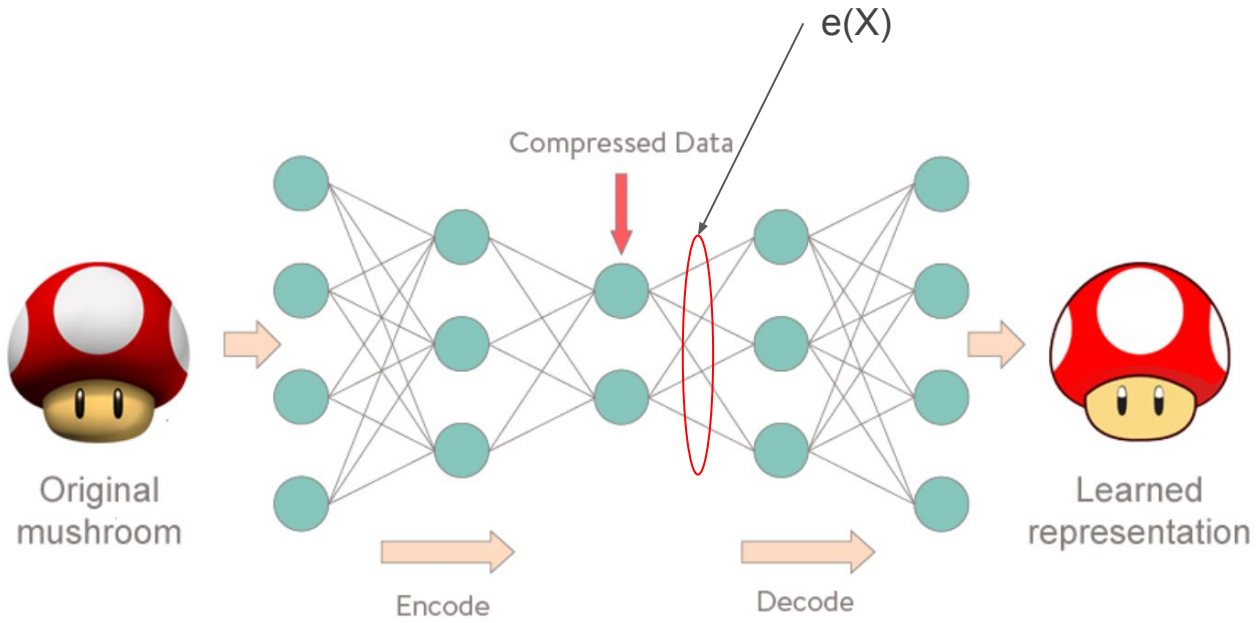
Важная идея: АК превращает объект X...

... на входе НС в числовой вектор e(X) — вектор значений, выходящих из нейронов бутылочного горлышка.



Важная идея: АК превращает объект X...

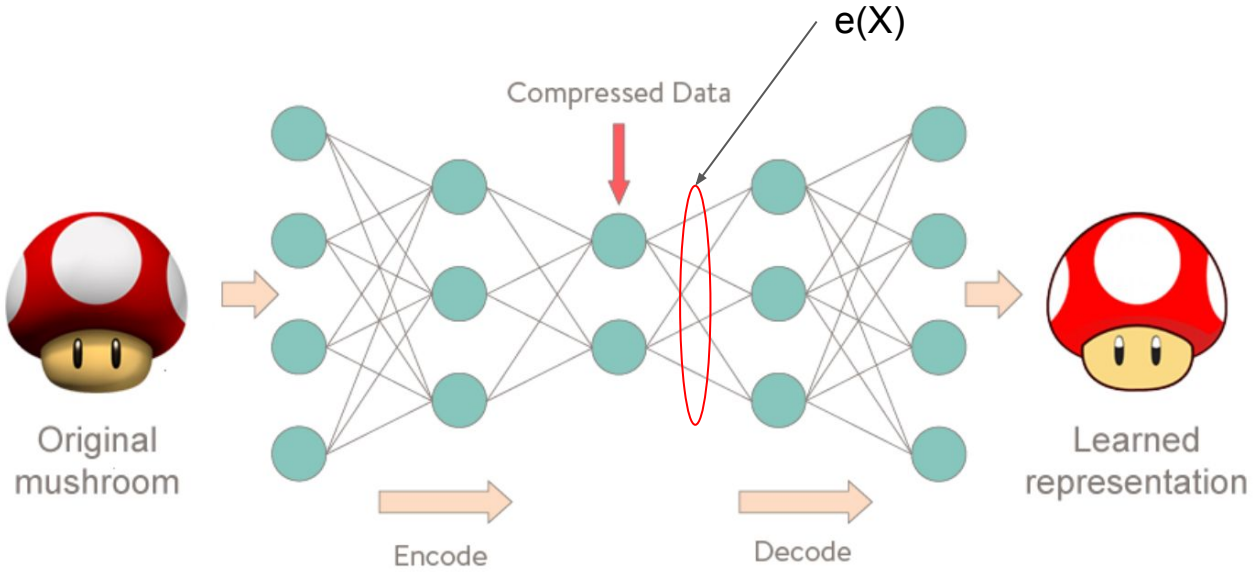
То есть, если бутылочное горлышко состоит из N нейронов, то объект X будет представлен в виде **вектора e(X) длины N**.



Важная идея: АК превращает объект X...

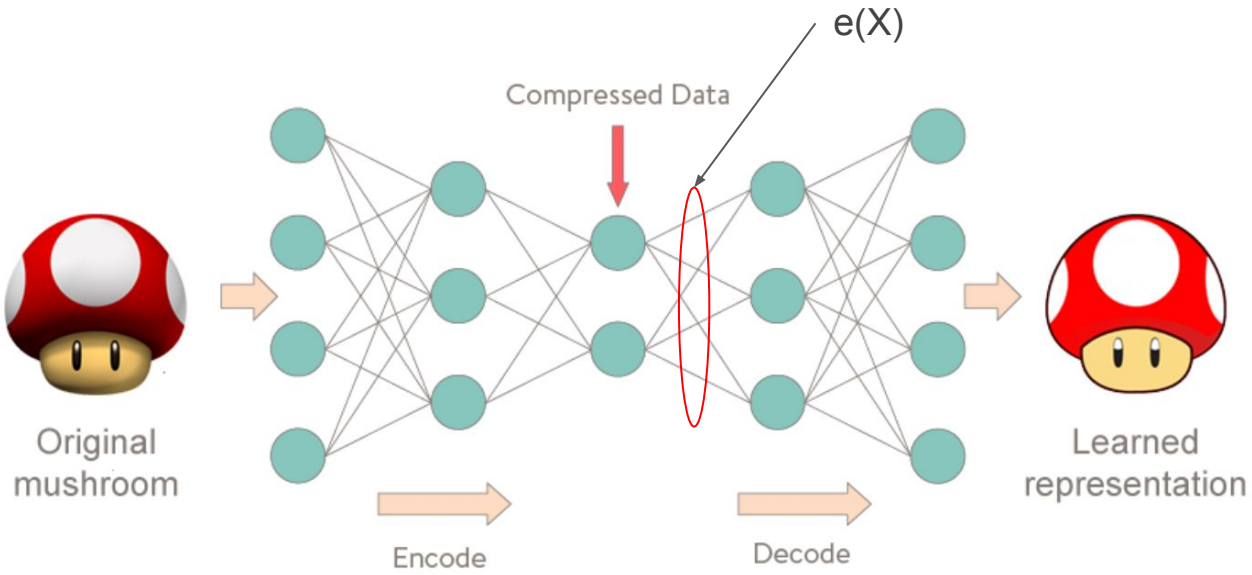
Идея превращать сложные объекты в числовые вектора очень важна в ML.

Это называется **embedding** (или числовым представлением).



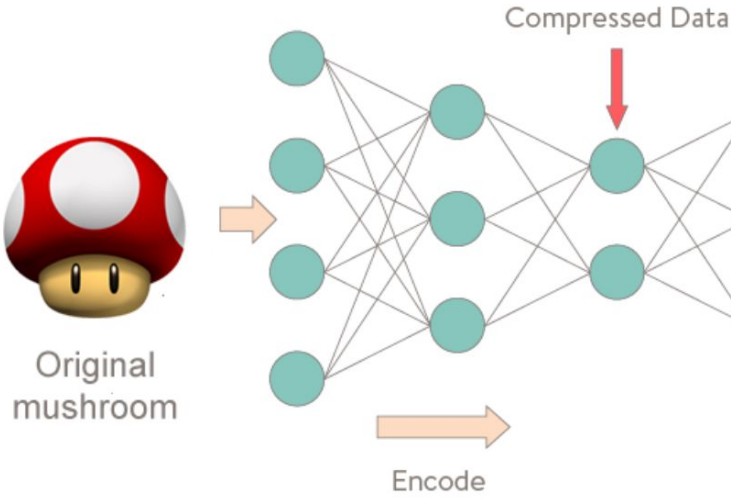
Важная идея: АК превращает объект X...

Embedding с помощью АК обладает одним **важным свойством:** похожие объекты X_1, X_2 будут закодированы близкими друг к другу векторами $e(X_1)$, $e(X_2)$.



Первое применение АК: архиватор

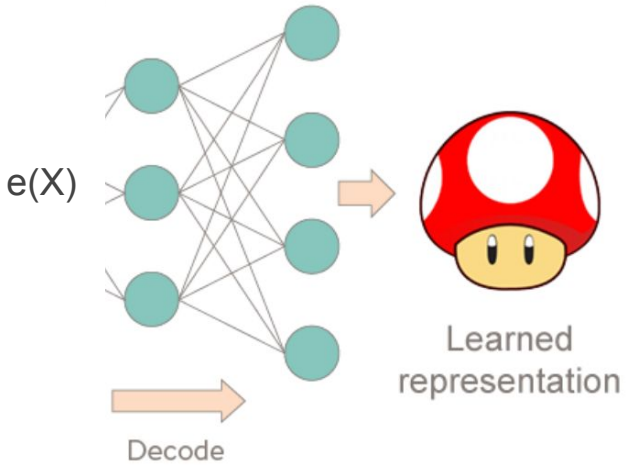
Объект X превращаем в вектор e(X) и храним его вместо объекта X.



e(X)

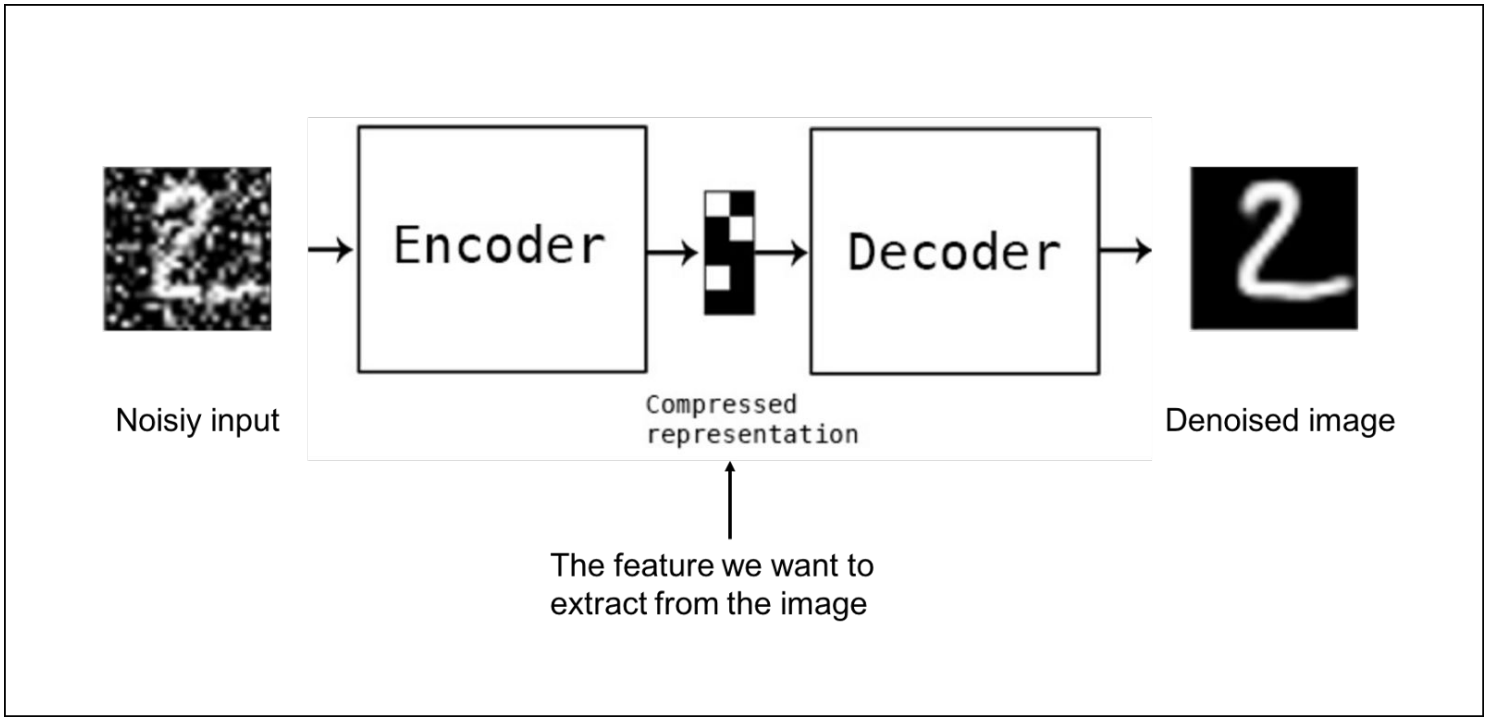
Первое применение АК: архиватор

Когда нам нужно восстановить объект X, то мы подаём вектор e(X) декодирующей части АК:



Второе применение: подавление шумов

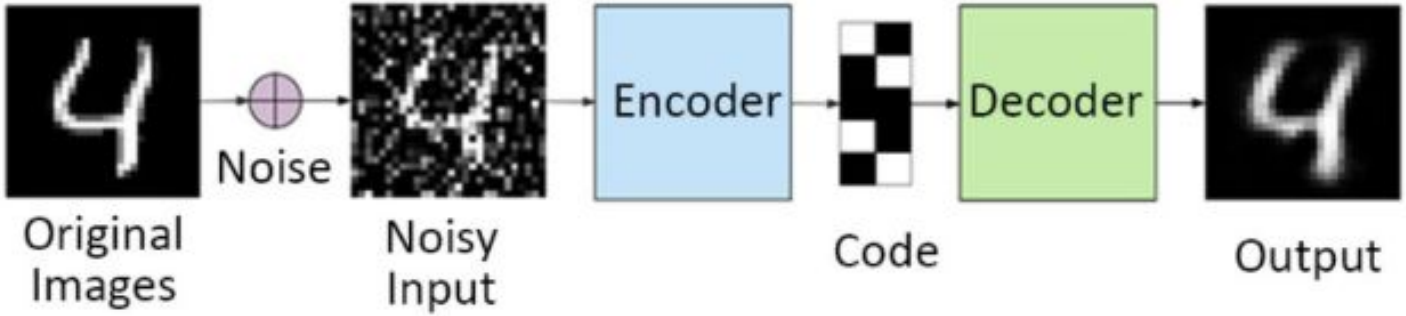
Изображение нужно очистить от шума. Но как натренировать АК?



Второе применение: подавление шумов

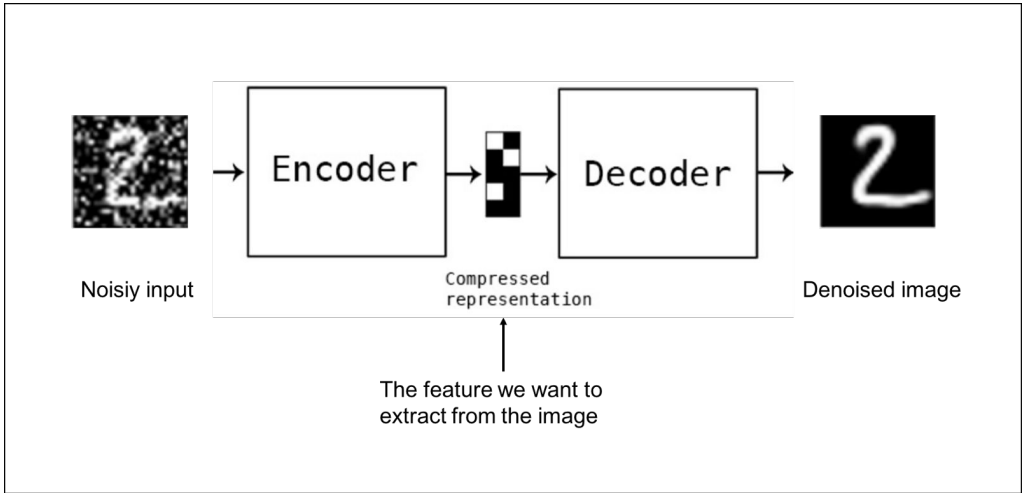
Тренировка АК: берём изображение X, генерируем случайный шум, получаем зашумлённое изображение X’.

Тренируем АК на парах (X’,X).



Второе применение: подавление шумов

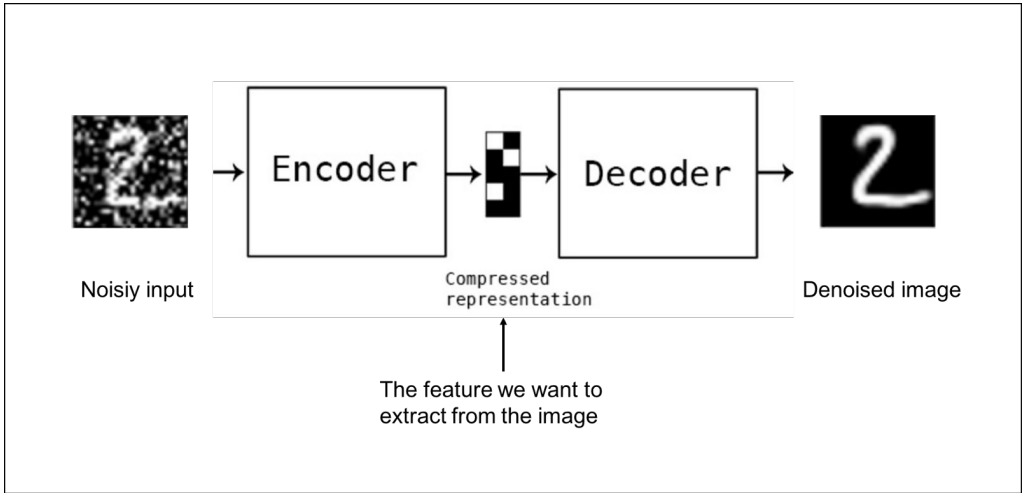
Итак, АК натренирован. Теперь ему на вход поступает зашумлённое изображение с неизвестным ответом. Мы его просто пропускаем через АК, и на выходе будет очищенное от шума изображение.



Второе применение: подавление шумов

Почему это работает?

АК понимает, что протащить всю информацию об изображении через бутылочное горлышко не получится, и поэтому через горлышко проходят только самые существенные части изображения, а всё случайное (шум) — отсеивается.



Третье применение: помощь в классификации

В задачах классификации НС необходимо предсказать **метку класса** у объекта. Грубо говоря, НС пытается угадать, что изображено на картинке. Такая НС тренируется на размеченной (тренировочной) выборке.

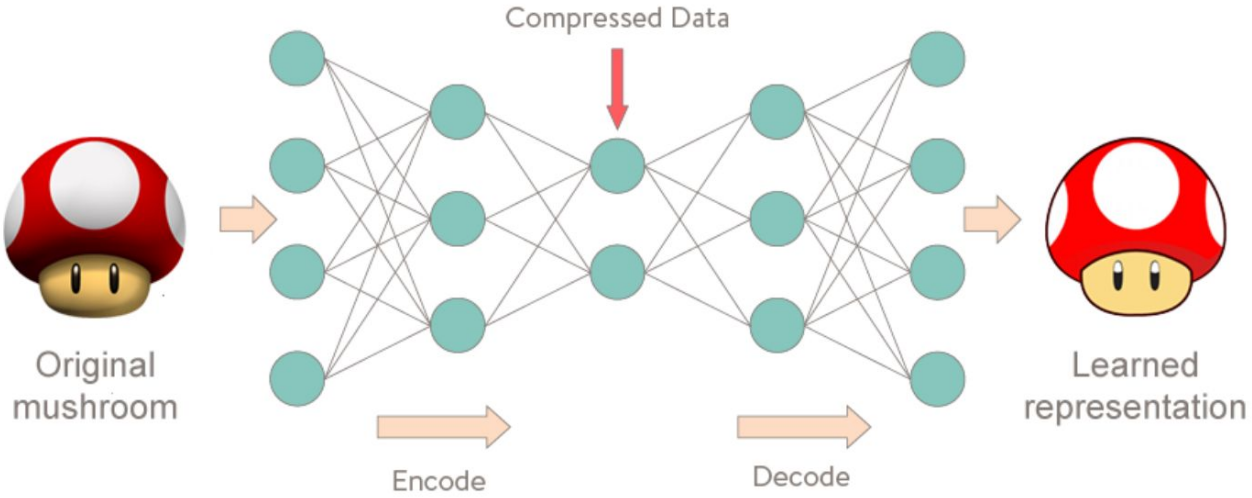
Размеченная выборка — это выборка объектов, для которых известен их точный класс. К сожалению, эта выборка может быть недостаточно хорошей по следующим причинам:

- **Размер выборки** — объектов в выборке слишком мало для качественного обучения НС
- **Дорогостоящая разметка** — разметить выборку крайне тяжело, так как для этого необходимо разметить достаточно много качественных исходных данных (к примеру, рентгеновских снимков) и найти специалистов, которые корректно сделают разметку (например, высококвалифицированных рентгенологов).

Как тут помогут АК?

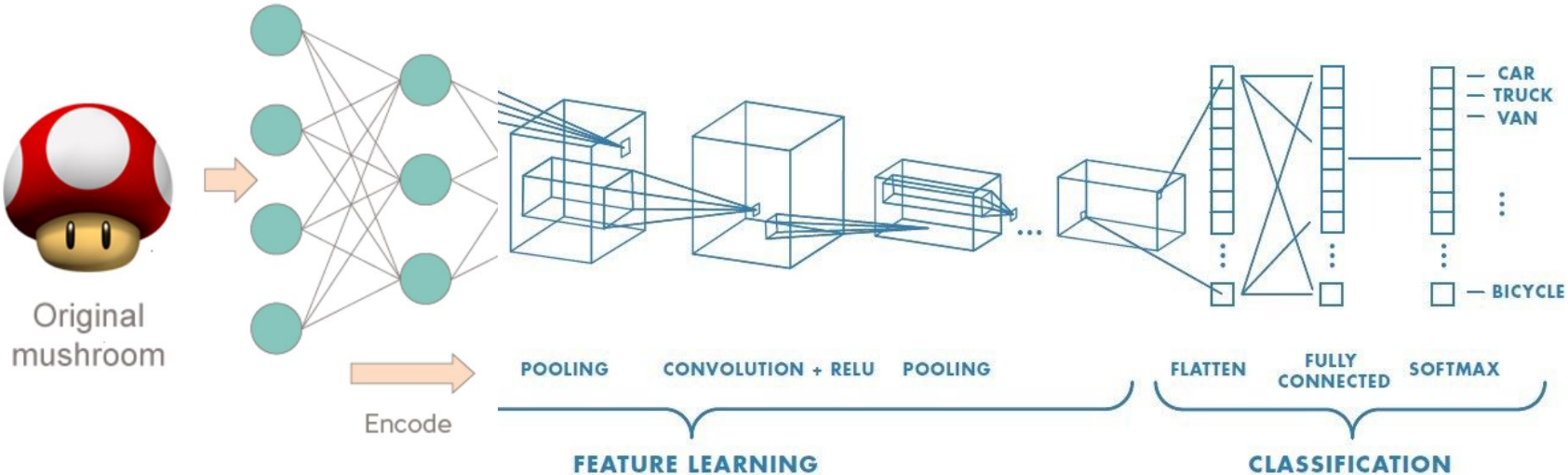
Третье применение: помощь в классификации

Пусть у нас небольшая размеченная выборка, но много неразмеченных картинок. Пусть $P_1,...,P_n$ — множество неразмеченных картинок.
Мы по парам $(P_1,P_1),...,(P_n,P_n)$ тренируем АК.



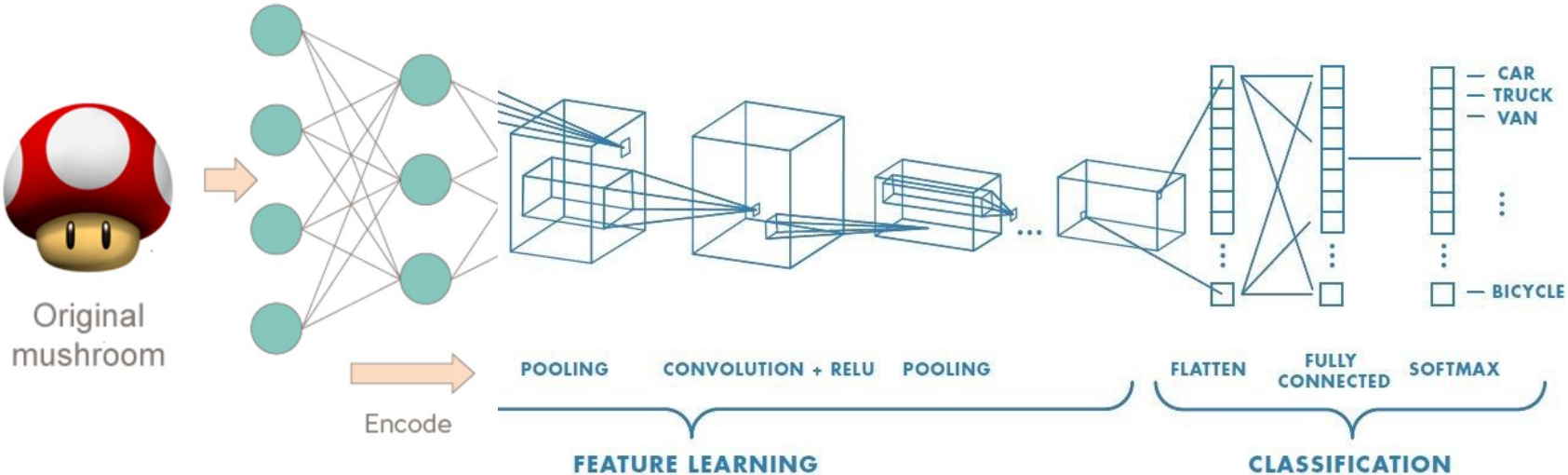
Третье применение: помощь в классификации

После этого оставляем от АК кодирующие слои и добавляем новые слои для распознавания изображений.
Получается такая сеть-монстр.



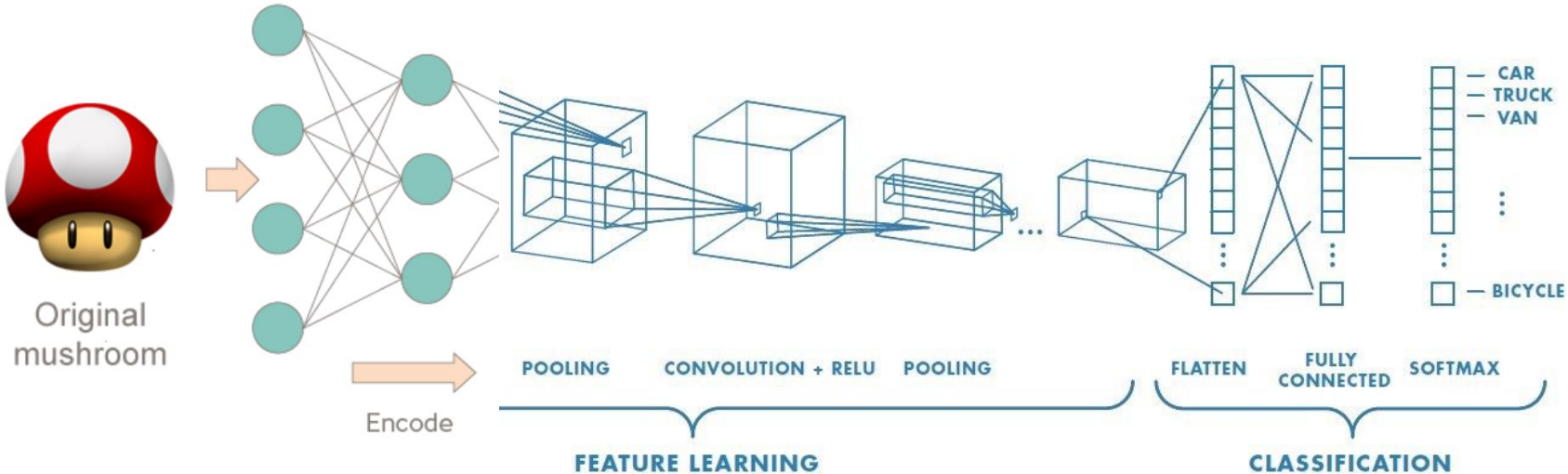
Третье применение: помощь в классификации

Теперь по размеченной выборке тренируем сеть-монстра.
Важно: веса между нейронами АК «заморожены», то есть они уже **не меняются.**



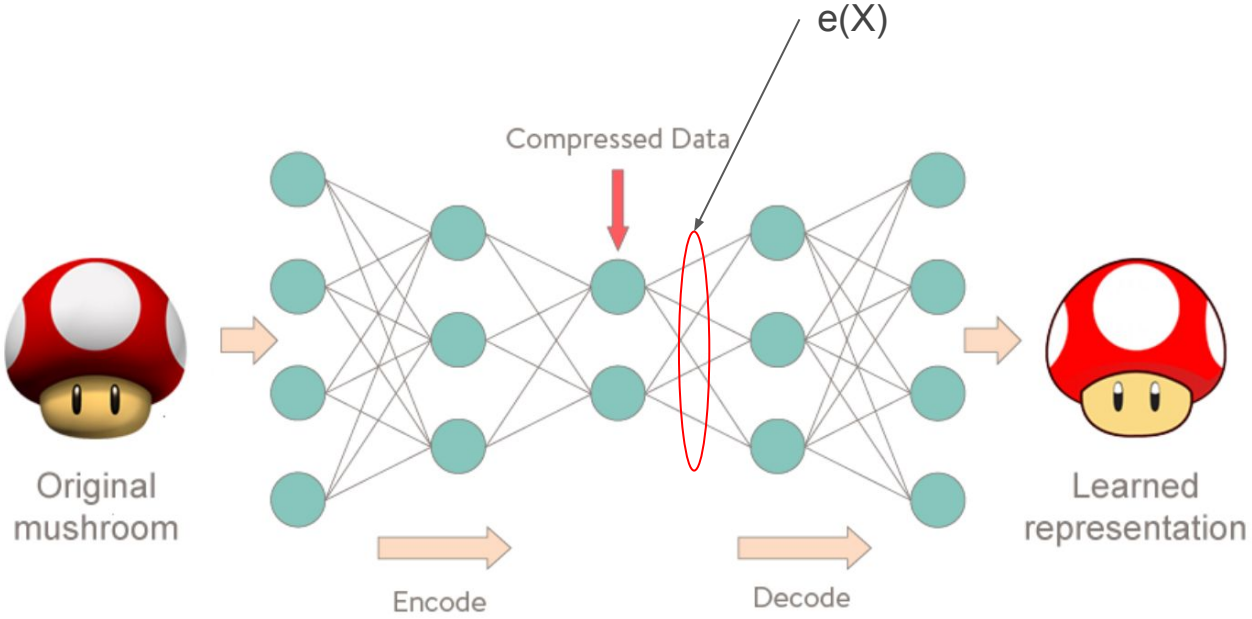
Третье применение: помощь в классификации

АК реально помогают решать задачу классификации. Можно иметь порядка тысяч неразмеченных изображений и несколько десятков размеченных. И тем не менее данный подход позволит правильно всё классифицировать.



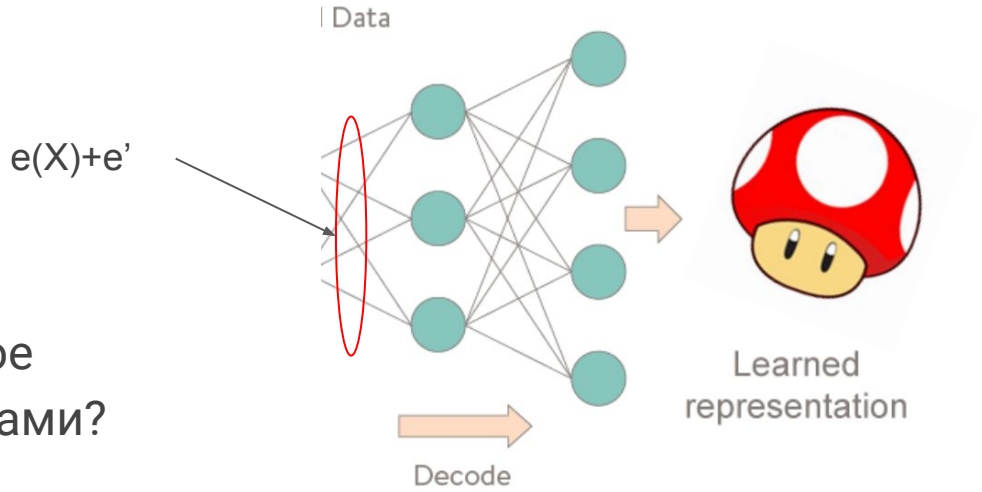
Четвёртое применение: генерация новых изображений

Мы знаем, что бутылочное горлышко АК изображение X превращает в вектор e(X). А потом вектор e(X) проходит через слой декодирования и получается похожее на X изображение.



Четвёртое применение: генерация новых изображений

Мы можем слегка менять вектор e(X) перед отправкой его в декодирующую часть (подаём вектор e(X)+e’). В этом случае и итоговое изображение будет отлично от X. Степень похожести сгенерированного изображения на оригинал X определяется величиной сдвига e’.



А вот как менять вектор e(X) осмысленно, чтобы получалось новое изображение с требуемыми свойствами?

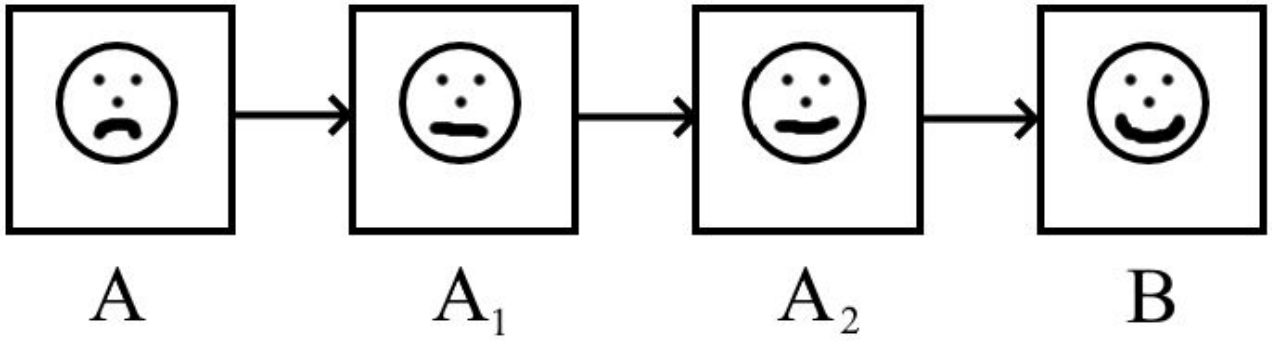
Генерация промежуточных изображений

Если бутылочное горлышко АК превращает изображения A,B в вектора e(A), e(B), то для генерации промежуточного изображения можно взять вектор «между ними».

Промежуточный вектор задаётся формулой:

$$\alpha e(A)+(1-\alpha)e(B),$$

где α — число из интервала (0,1).

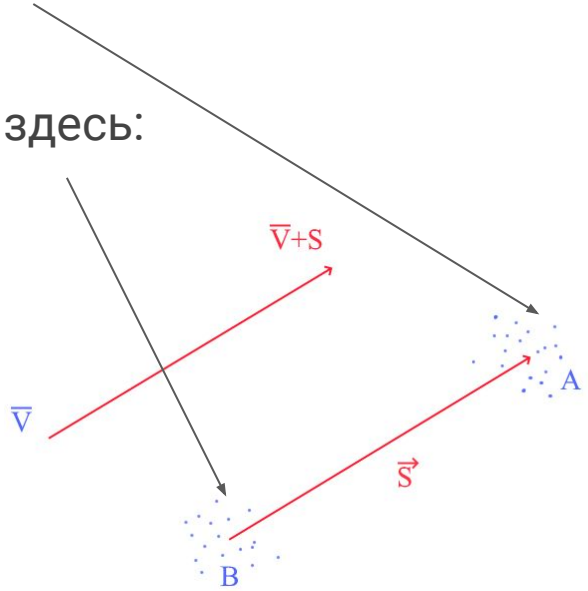


Выделение отдельных признаков

Допустим, что мы хотим добавлять улыбку (очки, бороду...) на фото людей.

Для этого мы прогоним через АК фото всех улыбающихся людей и получим соответствующие им вектора e(X). **Вот это облако точек**

Прогоним через АК фото грустных людей.
Соответствующее им **облако векторов e(X)** находится здесь:

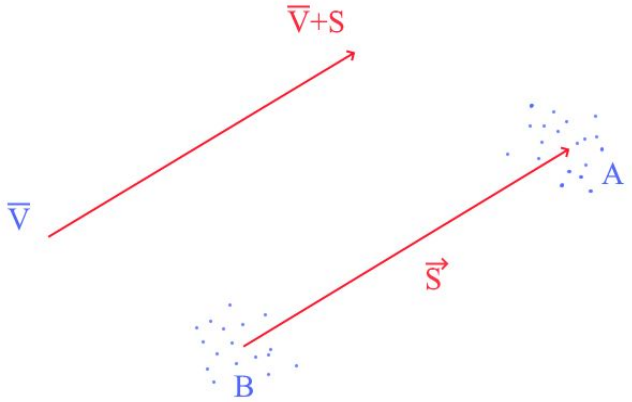


Найдём центр у каждого облака. Соединим центры вектором S.
Ну вот и все:
вектор S — вектор, отвечающий за улыбку.

Как им пользоваться?

Выделение отдельных признаков

Пусть нам надо наложить улыбку на грустное фото. Бутылочное горлышко АК превращает это фото в вектор e(X).
Добавляем к нему вектор S. Оказывается, что вектор e(X)+S при пропуске через декодирующую часть АК даст фото того же самого человека, но с улыбкой.



Выводы

- Мы рассмотрели архитектуру автокодировщика и показали процесс его тренировки.
- Было рассмотрено несколько приложений автокодировщиков в задачах анализа данных.
- Были рассмотрены методы генерации новых изображений и переноса стиля с помощью автокодировщиков.