Домашнее задание 2.

The file should be sent in the `.pdf` format created via $\LaTeX$ or typora or printed from pdf with the colab\jupyter notebook. The only handwritten part, that could be included in the solution are the figures and illustrations.

Deadline: 08.05.22 21:59:59

😱 Newton convergence issue

Рассмотрите следующую функцию:

$$
f(x, y) = \frac{x^4}{4} - x^2 + 2x + (y-1)^2
$$

И точку старта $x_0 = (0, 2)^\top$. Как ведет себя метод Ньютона, запущенный с этой точки? Чем это можно объяснить?

Как ведет себя градиентный спуск с фиксированным шагом $\alpha = 0.01$ и метод наискорейшего спуска в таких же условиях? (в этом задании не обязательно показывать численные симуляции)

1. Метод Ньютона для функции двух переменных расщипляется на два независимых метода Ньютона для одномерных функций: $x_{k+1} = x_k - \frac{f_1'(x)}{f_1''(x)}, \quad y_{k+1} = y_k - \frac{f_1'(y)}{f_1''(y)}$ Так как функция $f_2(y) = (y-1)^2$ квадратичная, то метод Ньютона по сойдется за одну итерацию. $f_1(x) не выпукла, поэтому гарантий сходимости метода Ньютона вообще нет. В данной начальной точке функция вогнута, и оказывается, что по оси x метод Ньютона будет прыгать из 0 в 1 и обратно.

Quasi Comparison

> Блок с отступами

Реализуйте на языке python:

- метод Ньютона
- метод SR-1

для минимизации следующих функций:

- Квадратичная форма $f(x) = \frac{1}{2}x^\top Ax + b^\top x, \quad x \in \mathbb{R}^n, A \in \mathbb{S}^{n \times n}_+$. Попробуйте n = 2, 50, 228
- Функция Розенброка $f(x, y) = (1-x)^2 + 100(y - x^2)^2$.

Сравните 2 реализованных Вами метода И метод `BFGS` из библиотеки `scipy`, а так же его модификацию L-BFGS в решении задачи минимизации описанных выше функций. точку старта необходимо инициализировать одинаковую для всех методов в рамках одного запуска. Необходимо провести не менее 10 запусков для каждого метода на каждой функции до достижения того критерия остановки, который вы выберете (например, расстояние до точки оптимума - во всех задачах мы её знаем)

В качестве результата нужно заполнить следующие таблички, заполнив в них усредненное по числу запусков количество итераций, необходимых для сходимости и времени работы:

Критерий остановки $||x_k - x^*|| < \epsilon$

Число запусков `20`, согласно подсчётам

P.S. если в силу каких то причин Вам не удалось сделать задание полностью, попробуйте сфокусироваться хотя бы на его части.

Квадратичная форма. n = 2	**Iterations**	**Time**
Newton	1	3.79e-05
SR-1	2	8.36e-05
BFGS	7	0.00074
L-BFGS	7	0.00041

Квадратичная форма. n = 50	**Iterations**	**Time**
Newton	1	0.00014
SR-1	2	0.00021
BFGS	49	0.05502
L-BFGS	33	0.03715

Квадратичная форма. n = 228 (Серьёзно?)	**Iterations**	**Time**
Newton	1	0.00131
SR-1	2	0.00179
BFGS	313	4.77063
L-BFGS	59	0.50464

Функция Розенброка	**Iterations**	**Time**
Newton	4	0.00019
SR-1	116	0.00639
BFGS	40	0.00332
L-BFGS	46	0.00127

```
def newton_descent(F, dF, ddF, w0, minima, epsilon = 0.01, Nsteps_max = 10000):
  prev = w0
  trajectory = [w0]
  n = 0
  while(np.linalg.norm(np.array(prev) - np.array(minima)) > epsilon):
    w = prev - np.linalg.inv(ddF(prev))@(dF(prev))
    prev = w
    trajectory.append(w)
    n+=1
    if (n >= Nsteps_max):
      print("Reached maximum number of steps")
      break
  return [np.array(trajectory), n]

def SR1(F, dF, ddF, w0, minima, epsilon = 0.01, Nsteps_max = 10000):
  prev = w0
  trajectory = [w0]
  B_prev = np.linalg.inv(ddF(prev))
  n = 0
  while(np.linalg.norm(np.array(prev) - np.array(minima)) > epsilon):
    if (n == 0):
      w = prev - B_prev@dF(prev)
      trajectory.append(w)
    else:
      B = B_prev + (((w - prev) - B_prev@(dF(w) - dF(prev))).reshape(((w - prev) - B_prev@(dF(w) - dF(prev))).shape[0], 1)@((w - prev) - B_prev@(dF(w) - dF(prev))).reshape(((w - prev) - B_prev@(dF(w) - d
      prev = w
      B_prev = B
      w = prev - B@dF(prev)
      trajectory.append(w)
```

```
    n+=1
    if (n >= Nsteps_max):
      print("Reached maximum number of steps")
      break
  return [np.array(trajectory), n]

import time
from scipy.optimize import minimize
from sklearn.datasets import make_spd_matrix
def QF(x):
  return (0.5*x.reshape((1, x.shape[0]))@A@x.reshape((x.shape[0], 1))).squeeze() + b@x
def dQF(x):
  return A@x + b
def ddQF(x):
  return A
N_launches = 20
for n in [2, 50, 228]:
  for i in range(N_launches):
    A = make_spd_matrix(n)
    b = np.random.randn(n)
    w0 = np.random.randn(n)
    minima = -np.linalg.inv(A)@b
    Newton_n = []
    Newton_time = []
    SR1_n = []
    SR1_time = []
    BFGS_n = []
    BFGS_time = []
    LBFGS_n = []
    LBFGS_time = []

    start_time = time.time()
    Newton_n.append(newton_descent(QF, dQF, ddQF, w0, minima = minima)[1])
    Newton_time.append(time.time() - start_time)

    start_time = time.time()
    SR1_n.append(SR1(QF, dQF, ddQF, w0, minima = minima)[1])
    SR1_time.append(time.time() - start_time)

    start_time = time.time()
    BFGS_n.append(minimize(QF, w0, method='BFGS').nit)
    BFGS_time.append(time.time() - start_time)

    start_time = time.time()
    LBFGS_n.append(minimize(QF, w0, method='L-BFGS-B').nit)
    LBFGS_time.append(time.time() - start_time)

    print("n = " + str(n))
    print("BFGS num of iterations = " + str(int(np.mean(BFGS_n))))
    print("BFGS avg time(s) = " + str(np.mean(BFGS_time)))
    print('\n')
    print("LBFGS num of iterations = " + str(int(np.mean(LBFGS_n))))
    print("LBFGS avg time(s) = " + str(np.mean(LBFGS_time)))
    print('\n')
    print("newton num of iterations = " + str(int(np.mean(Newton_n))))
    print("newton avg time(s) = " + str(np.mean(Newton_time)))
    print('\n')
    print("SR1 num of iterations = " + str(int(np.mean(SR1_n))))
```

```
    print("SR1 avg time(s) = " + str(np.mean(SR1_time)))
    print("\n==================================================\n")
```

```
LBFGS num of iterations = 36
LBFGS avg time(s) = 0.05391407012939453


newton num of iterations = 1
newton avg time(s) = 0.0011410713195800781


SR1 num of iterations = 2
SR1 avg time(s) = 0.0010385513305664062

==================================================

/usr/local/lib/python3.7/dist-packages/scipy/optimize/optimize.py:1058: RuntimeWarning: divide by zero encountered in double_scalars
  rhok = 1.0 / (numpy.dot(yk, sk))
n = 50
BFGS num of iterations = 106
BFGS avg time(s) = 0.45835423469543457


LBFGS num of iterations = 71
LBFGS avg time(s) = 0.1068274974822998


newton num of iterations = 1
newton avg time(s) = 0.0005221366882324219


SR1 num of iterations = 2
SR1 avg time(s) = 0.0005354881286621094

==================================================

n = 50
BFGS num of iterations = 74
BFGS avg time(s) = 0.2761204242706299


LBFGS num of iterations = 86
LBFGS avg time(s) = 0.13574934005737305


newton num of iterations = 1
newton avg time(s) = 0.0005114078521728516


SR1 num of iterations = 2
SR1 avg time(s) = 0.0005290508270263672

==================================================

n = 50
BFGS num of iterations = 52
BFGS avg time(s) = 0.1508331298828125


LBFGS num of iterations = 88
LBFGS avg time(s) = 0.1415388584136963
```

```
def Rosen(x):
```

```
  return (1 - x[0])**2 + 100*(x[1] - x[0]**2)**2
def dRosen(x, y):
  dR1 = 2*(x - 1) + 400*x*(x**2 - y)
  dR2 = 200*(y - x**2)
  return np.array([dR1, dR2])
def ddRosen(x, y):
  ddR = np.zeros((2, 2))
  ddR[0][0] = 2 + 400*(x**2 - y) + 800*x**2
  ddR[0][1] = -400*x
  ddR[1][0] = -400*x
  ddR[1][1] = 200
  return np.array(ddR)
def newton_descent(F, dF, ddF, w0, minima, epsilon = 0.01, Nsteps_max = 10000):
  prev = w0
  trajectory = [w0]
  n = 0
  while(np.linalg.norm(np.array(prev) - np.array(minima)) > epsilon):
    w = prev - np.linalg.inv(ddF(prev[0], prev[1]))@(dF(prev[0], prev[1]))
    prev = w
    trajectory.append(w)
    n+=1
    if (n >= Nsteps_max):
      print("Reached maximum number of steps")
      break
  return [np.array(trajectory), n]
def SR1(F, dF, ddF, w0, minima, epsilon = 0.01, Nsteps_max = 10000):
  prev = w0
  trajectory = [w0]
  B_prev = np.linalg.inv(ddF(prev[0], prev[1]))
  n = 0
  while(np.linalg.norm(np.array(prev) - np.array(minima)) > epsilon):
    if (n == 0):
      w = prev - B_prev@dF(prev[0], prev[1])
      trajectory.append(w)
    else:
      B = B_prev + (((w - prev) - B_prev@(dF(w[0], w[1]) - dF(prev[0], prev[1]))).reshape(2, 1)@((w - prev) - B_prev@(dF(w[0], w[1]) - dF(prev[0],prev[1]))).reshape(2, 1).T)/((w - prev - B_prev@(dF(w[0],
      prev = w
      B_prev = B
      w = prev - B@dF(prev[0], prev[1])
      trajectory.append(w)
    n+=1
    if (n >= Nsteps_max):
      print("Reached maximum number of steps")
      break
  return [np.array(trajectory), n]

import time
import numpy as np

from scipy.optimize import minimize
N_launches = 20

w0 = np.array([3.22, 13.37])
minima = [1, 1]
Newton_n = []
Newton_time = []
SR1_n = []
SR1_time = []
```

```
BFGS_n = []
BFGS_time = []
LBFGS_n = []
LBFGS_time = []
for i in range(N_launches):

  start_time = time.time()
  Newton_n.append(newton_descent(Rosen, dRosen, ddRosen, w0, minima = minima)[1])
  Newton_time.append(time.time() - start_time)

  start_time = time.time()
  SR1_n.append(SR1(Rosen, dRosen, ddRosen, w0, minima = minima)[1])
  SR1_time.append(time.time() - start_time)

  start_time = time.time()
  BFGS_n.append(minimize(Rosen, w0, method='BFGS').nit)
  BFGS_time.append(time.time() - start_time)

  start_time = time.time()
  LBFGS_n.append(minimize(Rosen, w0, method='L-BFGS-B').nit)
  LBFGS_time.append(time.time() - start_time)

print("BFGS num of iterations = " + str(int(np.mean(BFGS_n))))
print("BFGS avg time(s) = " + str(np.mean(BFGS_time)))
print('\n')
print("LBFGS num of iterations = " + str(int(np.mean(LBFGS_n))))
print("LBFGS avg time(s) = " + str(np.mean(LBFGS_time)))
print('\n')
print("newton num of iterations = " + str(int(np.mean(Newton_n))))
print("newton avg time(s) = " + str(np.mean(Newton_time)))
print('\n')
print("SR1 num of iterations = " + str(int(np.mean(SR1_n))))
print("SR1 avg time(s) = " + str(np.mean(SR1_time)))
```

```
BFGS num of iterations = 40
BFGS avg time(s) = 0.007977175712585449


LBFGS num of iterations = 46
LBFGS avg time(s) = 0.0037010908126831055


newton num of iterations = 4
newton avg time(s) = 0.0013116359710693359


SR1 num of iterations = 225
SR1 avg time(s) = 0.02566990852355957
```

😺 Conjugate gradients with preconditioner

Метод

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
if $\mathbf{r}_0$ is sufficiently small, then return $\mathbf{x}_0$ as the result
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat
$\quad\alpha_k := \dfrac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top \mathbf{A}\mathbf{p}_k}$
$\quad\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
$\quad\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$
$\quad$if $\mathbf{r}_{k+1}$ is sufficiently small, then exit loop
$\quad\beta_k := \dfrac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}$
$\quad\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
$\quad k := k+1$
end repeat
return $\mathbf{x}_{k+1}$ as the result

В этом задании Вам предлагается рассмотреть как влияют предобуславливатели на время работы метода сопряженных градиентов.

Рассмотрим задачу минимизации квадратичной функции:

$$f(x) = \frac{1}{2}x^\top Ax - b^\top x$$

где $A \in \mathbb{S}^n_{++}, b \in \mathbb{R}^n$.

Как мы знаем, эта задача выпукла и минимум находится из условия $\nabla f(x^*) = Ax^* - b = 0$. То есть для решения задачи необходимо разрешить систему уравнений $Ax = b$. Можно просто применить метод сопряженных градиентов, но если матрица плохо обусловлена ($\frac{\lambda_{max}}{\lambda_{min}} >> 1$), метод работает медленно (буквально, скорость сходимости CG прямо пропорциональна $\sqrt{\kappa(A)}$).

Preconditioning

Один из способов борьбы с этим - использование матриц-предобуславливателей разных видов и последующее решение другой задачи:

$$MAx = Mb$$

Здесь матрица **предобуславливателя** M подбирается таким образом, чтобы итоговая матрица $\tilde{A} = MA$ имела меньшее число обусловленности. Существует несколько довольно простых, но зачастую сильно улучшающих работу метода предоубславливателей:

- $M = A^{-1}$ (Ideal preconditioner)
- $M = \text{diag}(A_{11}^{-1}, A_{22}^{-1}, \dots, A_{nn}^{-1})$ (Jacobi)
- $M \approx \hat{A}$, где например $\hat{A}$ - неполная факторизация Холецкого

Preconditioned Conjugate Gradients

Лучшая ссылка - с.39. Нет никаких проблем в том, чтобы решать новую систему $\tilde{A}x = \tilde{b}$ методов сопряженных градиентов. Однако, нативное встраивание предобуславливателя в алгоритм, делает использование этой идеи еще более эффективной. Для этого надо детально модифицировать классический CG. Кроме того, мы потребуем положительности новой матрицы $\tilde{A}$. Для этого будем использовать следующий вариант построения матрицы M:

$$M^{-1} = LL^\top$$

$$\begin{aligned} Ax = b &\leftrightarrow M^{-1}Ax = M^{-1}b \\ &\leftrightarrow L^\top Ax = L^\top b \\ &\leftrightarrow \underbrace{L^\top AL}_{\tilde{A}} \cdot \underbrace{L^{-1}x}_{\tilde{x}} = \underbrace{L^\top b}_{\tilde{b}} \end{aligned}$$

В новых переменных $(\tilde{A}, \tilde{x}, \tilde{b})$ невязка запишется, как:

$$\tilde{r}_k = \tilde{b} - \tilde{A}\tilde{x}_k = L^\top b - (L^\top AL)(L^{-1}x_k) = L^\top b - L^\top Ax_k = L^\top r_k$$

Факторизация Холецкого s.p.d. матрицы A - ее разложение на произведение нижнетреугольной и верхнетреугольной матрицы: $A = L^T L$ wiki. Есть несколько упрощений этого алгоритма, позволяющих получить матрицу, "похожую" на A. Мы будем использовать следующую: $if \quad (a_{i,j} = 0) \to l_{i,j} = 0$, а далее по алгоритму.

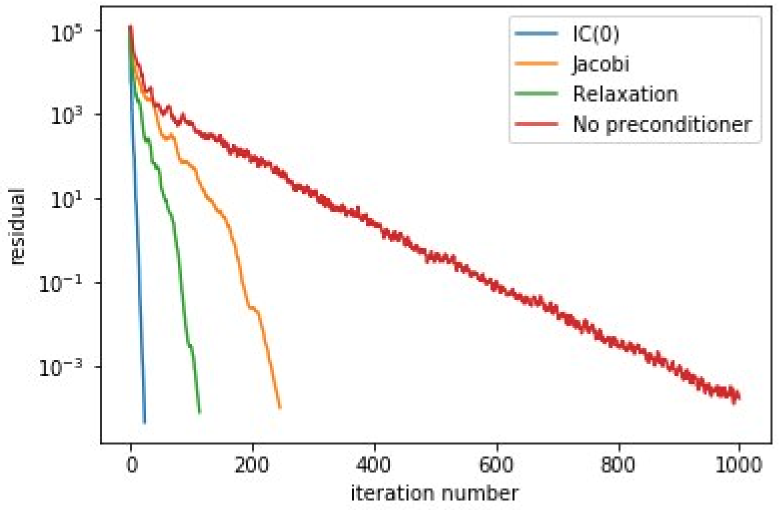
Задание Выбрать 1 задачу отсюда (выбирайте формат matrix market - его умеет читать scipy), исследовать как влияет на скорость сходимости тот или иной предоубславливатель:

1) Сравнить число итераций, за которое метод сходится с точностью 10^{-7} для двух предобуславливателей и для обычного метода сопряженных градиентов.

2) Построить графики зависимости нормы невязки $\|r_k\| = \|Ax_k - b\|$ от номера итерации для трех предобуславливателей и для обычного метода сопряженных градиентов. Обратите внимание, что в этом задании можно использовать дефолтный метод сопряженных градиентов из scipy - там есть возможность в качестве аргумента передать preconditioner.

3) Сравнить итоговое время работы методов до сходимости. Обратите внимание, что для честного сравнения по времени не стоит использовать дополнительных сложных callback-ов.

Пример:



==YOUR ANSWER==

```
from scipy.io import mmread
np.random.seed(10)
A = mmread("/content/sample_data/685_bus.mtx").toarray()
#mtx = mmread("/content/counts_unfiltered/cells_x_genes.mtx")
b = np.random.normal(0, 20, (A.shape[0],))
np.random.seed(11)
A = mmread('/content/sample_data/1138_bus.mtx').toarray()
b = np.random.normal(0, 20, (A.shape[0],))
```

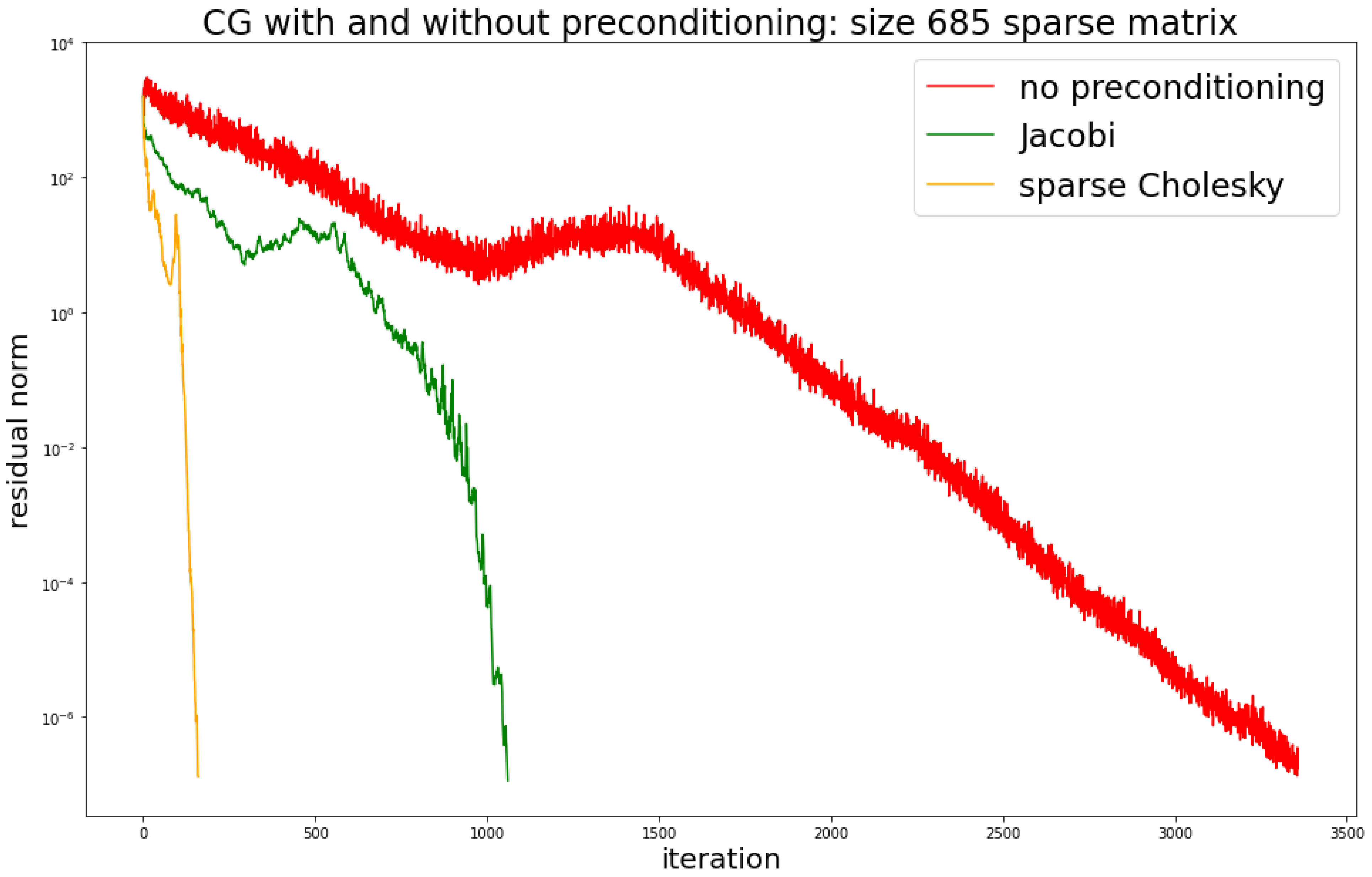
```
def chol(A):
 n = A.shape[0]
 L = np.zeros_like(A)
 for j in range(n):
  L[j,j] = np.sqrt(A[j,j] - (L*L)[j,:j].sum())
 for i in range(j, n):
  L[i,j] = 1/L[j,j] * (A[i,j] - (L[i,:j] * L[j,:j]).sum())
 return L
def chol_sparse(A):
 n = A.shape[0]
 L = np.zeros_like(A)
 for j in range(n):
  L[j,j] = np.sqrt(A[j,j] - (L*L)[j,:j].sum())
  for i in range(j, n):
    if A[i,j] == 0:
      L[i,j] = 0
      continue
    L[i,j] = 1/L[j,j] * (A[i,j] - (L[i,:j] * L[j,:j]).sum())
 return L


import numpy as np
def conjugate_grad(A, b, x=None, max_iter_mult=1.0, eps=1e-7):
 """
 Description
 -----------
 Solve a linear equation Ax = b with conjugate gradient method.
 Parameters
 ----------
 A: 2d numpy.array of positive semi-definite (symmetric) matrix
 b: 1d numpy.array
 x: 1d numpy.array of initial point
 Returns
 -------
 list of residuals
 """
 n = len(b)
 rks = []
 if x is None:
  x = np.ones(n)
 r = b - A @ x
 #print(r[:3])
 p = np.copy(r)
 r_k_norm = r.T @ r
 for i in range(int(max_iter_mult*n)):
  #print(i+1)
  rks.append(np.sqrt(r_k_norm))
  Ap = A @ p
  alpha = r_k_norm / (p @ Ap)
  x += alpha * p
  r -= alpha * Ap
  #print(i, r[:3])
  r_kplus1_norm = r @ r
  beta = r_kplus1_norm / r_k_norm
  #print(i, beta)
  r_k_norm = r_kplus1_norm
  if np.sqrt(r_kplus1_norm) < eps:
    print('Iterations:', i)
    break
```

```
  #print(i, r[0:2], beta, p[0:2])
  p = r + beta * p
  #print(i, p[:3])
 return x, rks
def preconditioned_conjugate_grad(A, b, M, x=None, max_iter_mult=1.0, eps=1e-7):
 """
 Description
 -----------
 Solve a linear equation Ax = b with conjugate gradient method.
 Parameters
 ----------
 A: 2d numpy.array of positive semi-definite (symmetric) matrix
 M: 2d numpy.array of positive semi-definite (symmetric) matrix - preconditio
ner (already inverted)
 b: 1d numpy.array
 x: 1d numpy.array of initial point
 Returns
 -------
 list of residuals
 """
 n = len(b)
 rks = []
 if x is None:
  x = np.ones(n)
 r = b - A @ x
 #print(r[:3])
 z = M @ r
 p = np.copy(z)
 r_k_norm = np.linalg.norm(r)
 r_k_z_k = r @ z
 for i in range(int(max_iter_mult*n)):
  #print(i+1)
  rks.append(r_k_norm)
  Ap = A @ p
  alpha = (r_k_z_k) / (p @ Ap)
  x += alpha * p
  r -= alpha * Ap
  #print(i, r[:3])
  r_kplus1_norm = np.linalg.norm(r)
  z = M @ r
  r_k1_z_k1 = r @ z
  beta = r_k1_z_k1 / r_k_z_k
  #print(i, beta)
  r_k_norm = r_kplus1_norm
  r_k_z_k = r_k1_z_k1
  if r_kplus1_norm < eps:
    print('Iterations:', i)
    break
  #print(i, z[0:2], beta, p[0:2])
  p = z + beta * p
  #print(i, p[:3])
 return x, rks

import scipy
x1, rks1 = conjugate_grad(A, b, x=np.ones_like(b), eps=1e-7, max_iter_mult=50)
M_jacobi = np.diag(1 / np.diag(A))
x2, rks2 = preconditioned_conjugate_grad(A, b, M_jacobi, x=np.ones_like(b), eps=
1e-7, max_iter_mult=50)
L = chol_sparse(A)
```

```
L_inv = scipy.linalg.solve_triangular(L, np.eye(A.shape[0]), lower=True)
M_chol = L_inv.T @ L_inv
x3, rks3 = preconditioned_conjugate_grad(A, b, M_chol, x=np.ones_like(b), eps=1e-7, max_iter_mult=50)
```

```
Iterations: 3358
Iterations: 1060
Iterations: 160
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(16,10))
plt.plot(rks1, color='red', label='no preconditioning')
plt.plot(rks2, color='green', label='Jacobi')
plt.plot(rks3, color='orange', label='sparse Cholesky')
plt.yscale('log')
plt.xlabel('iteration', fontsize=20)
plt.ylabel('residual norm', fontsize=20)
plt.title('CG with and without preconditioning: size 685 sparse matrix', fontsize=24)
plt.legend(fontsize=23)
#plt.savefig('task3_1138.png')
plt.show()
```



Возьмём 2 симметричные матрицы, которые положительно определены: матрица №1 размером 685 на 685 и с числом обусловленности $4.2 * 10^5$ и матрица №2 размером 1138 на 1138 с числом обусловленности $8, 6 * 10^6$. В первой матрице 0.69 процентов ненулевых элементов, а во второй 0.38 процентов. Почему взяты они? потому, что одна почти в 4 раза меньше другой. Сравним число итераций для достижения заданной точности невязки:

Итерации .... 1138_bus ..... 685_bus

no_preconditioning ... 3358 695

Jacobi 1060 260

(sparce) Cholesky 92 160

Stochastic optimization tricks

You will study stochastic optimization in the setting of timeseries anomaly detection using an autoencoder. source

Introduction

This script demonstrates how you can use a reconstruction convolutional autoencoder model to detect anomalies in timeseries data.

Setup

```
import numpy as np
import pandas as pd
from tensorflow import keras
from tensorflow.keras import layers
from matplotlib import pyplot as plt
```

Load the data

We will use the Numenta Anomaly Benchmark(NAB) dataset. It provides artifical timeseries data containing labeled anomalous periods of behavior. Data are ordered, timestamped, single-valued metrics.

We will use the `art_daily_small_noise.csv` file for training and the `art_daily_jumpsup.csv` file for testing. The simplicity of this dataset allows us to demonstrate anomaly detection effectively.

```
master_url_root = "https://raw.githubusercontent.com/numenta/NAB/master/data/"

df_small_noise_url_suffix = "artificialNoAnomaly/art_daily_small_noise.csv"
df_small_noise_url = master_url_root + df_small_noise_url_suffix
df_small_noise = pd.read_csv(
    df_small_noise_url, parse_dates=True, index_col="timestamp"
)

df_daily_jumpsup_url_suffix = "artificialWithAnomaly/art_daily_jumpsup.csv"
df_daily_jumpsup_url = master_url_root + df_daily_jumpsup_url_suffix
df_daily_jumpsup = pd.read_csv(
```

```
    df_daily_jumpsup_url, parse_dates=True, index_col="timestamp"
)
```

Quick look at the data

```
print(df_small_noise.head())

print(df_daily_jumpsup.head())
```

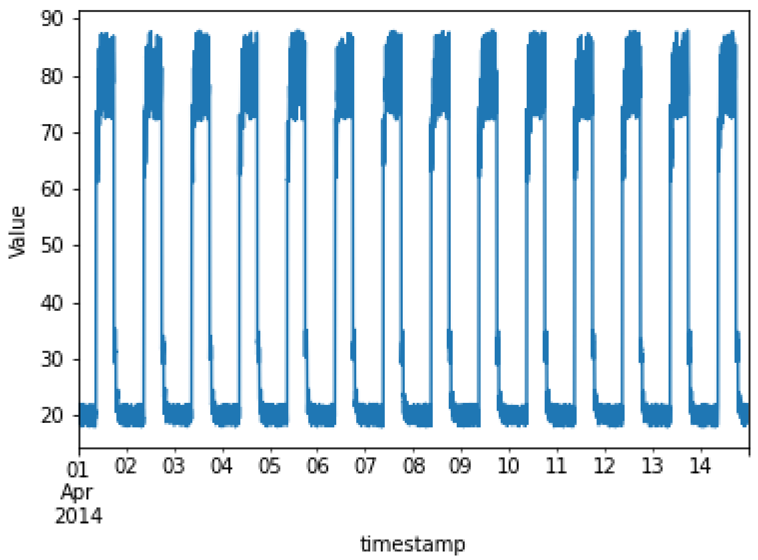
```
                         value
timestamp
2014-04-01 00:00:00  18.324919
2014-04-01 00:05:00  21.970327
2014-04-01 00:10:00  18.624806
2014-04-01 00:15:00  21.953684
2014-04-01 00:20:00  21.909120
                         value
timestamp
2014-04-01 00:00:00  19.761252
2014-04-01 00:05:00  20.500833
2014-04-01 00:10:00  19.961641
2014-04-01 00:15:00  21.490266
2014-04-01 00:20:00  20.187739
```

Visualize the data

Timeseries data without anomalies

We will use the following data for training.

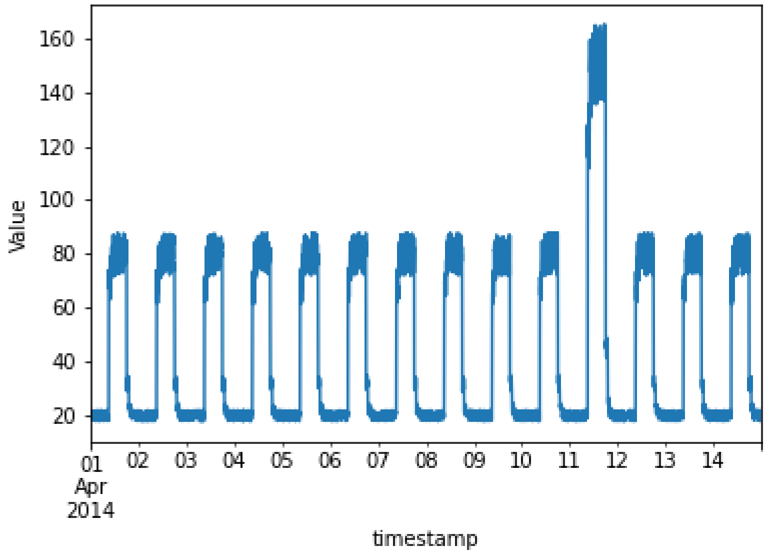
```
fig, ax = plt.subplots()
df_small_noise.plot(legend=False, ax=ax)
plt.ylabel('Value')
plt.show()
```



Timeseries data with anomalies

We will use the following data for testing and see if the sudden jump up in the data is detected as an anomaly.

```
fig, ax = plt.subplots()
df_daily_jumpsup.plot(legend=False, ax=ax)
plt.ylabel('Value')
plt.show()
```



Prepare training data

Get data values from the training timeseries data file and normalize the `value` data. We have a `value` for every 5 mins for 14 days.

- 24 * 60 / 5 = **288 timesteps per day**
- 288 * 14 = **4032 data points** in total

```
# Normalize and save the mean and std we get,
# for normalizing test data.
training_mean = df_small_noise.mean()
training_std = df_small_noise.std()
df_training_value = (df_small_noise - training_mean) / training_std
print("Number of training samples:", len(df_training_value))
```

```
Number of training samples: 4032
```

Create sequences

Create sequences combining `TIME_STEPS` contiguous data values from the training data.

```
TIME_STEPS = 288

# Generated training sequences for use in the model.
def create_sequences(values, time_steps=TIME_STEPS):
    output = []
    for i in range(len(values) - time_steps + 1):
        output.append(values[i : (i + time_steps)])
    return np.stack(output)
```

```
x_train = create_sequences(df_training_value.values)
print("Training input shape: ", x_train.shape)
```

```
Training input shape:  (3745, 288, 1)
```

Build a model

We will build a convolutional reconstruction autoencoder model. The model will take input of shape `(batch_size, sequence_length, num_features)` and return output of the same shape. In this case, `sequence_length` is 288 and `num_features` is 1.

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.01), loss="mse")
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv1d (Conv1D)             (None, 144, 32)           256       
                                                                 
 dropout (Dropout)           (None, 144, 32)           0         
                                                                 
 conv1d_1 (Conv1D)           (None, 72, 16)            3600      
                                                                 
 conv1d_transpose (Conv1DTra  (None, 144, 16)          1808      
 nspose)                                                         
                                                                 
 dropout_1 (Dropout)         (None, 144, 16)           0         
                                                                 
 conv1d_transpose_1 (Conv1DT  (None, 288, 32)          3616      
 ranspose)                                                       
                                                                 
 conv1d_transpose_2 (Conv1DT  (None, 288, 1)           225       
 ranspose)                                                       
                                                                 
=================================================================
Total params: 9,505
Trainable params: 9,505
```

```
Non-trainable params: 0
_________________________________________________________________
```

Train the model

Please note that we are using `x_train` as both the input and the target since this is a reconstruction model.

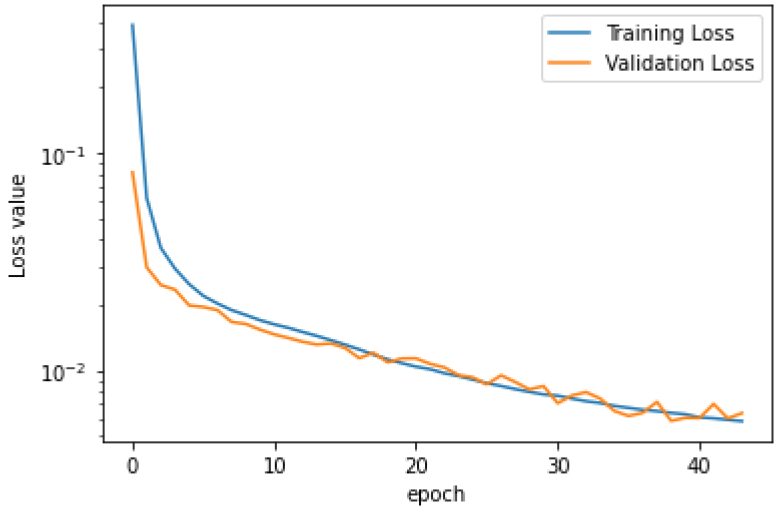
```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

```
Epoch 1/50
27/27 [==============================] - 4s 92ms/step - loss: 0.1939 - val_loss: 0.0419
Epoch 2/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0499 - val_loss: 0.0288
Epoch 3/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0339 - val_loss: 0.0231
Epoch 4/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0263 - val_loss: 0.0172
Epoch 5/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0218 - val_loss: 0.0151
Epoch 6/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0203 - val_loss: 0.0169
Epoch 7/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0183 - val_loss: 0.0171
Epoch 8/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0166 - val_loss: 0.0134
Epoch 9/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0153 - val_loss: 0.0157
Epoch 10/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0144 - val_loss: 0.0145
Epoch 11/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0132 - val_loss: 0.0157
Epoch 12/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0127 - val_loss: 0.0116
Epoch 13/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0112 - val_loss: 0.0102
Epoch 14/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0104 - val_loss: 0.0116
Epoch 15/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0092 - val_loss: 0.0096
Epoch 16/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0085 - val_loss: 0.0116
Epoch 17/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0077 - val_loss: 0.0103
Epoch 18/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0073 - val_loss: 0.0073
Epoch 19/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0069 - val_loss: 0.0094
Epoch 20/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0069 - val_loss: 0.0129
Epoch 21/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0065 - val_loss: 0.0090
```

```
Epoch 22/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0059 - val_loss: 0.0087
Epoch 23/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0058 - val_loss: 0.0055
Epoch 24/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0057 - val_loss: 0.0063
Epoch 25/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0055 - val_loss: 0.0082
Epoch 26/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0054 - val_loss: 0.0064
Epoch 27/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0052 - val_loss: 0.0137
Epoch 28/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0054 - val_loss: 0.0068
```

Let's plot training and validation loss to see how the training went.

```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```



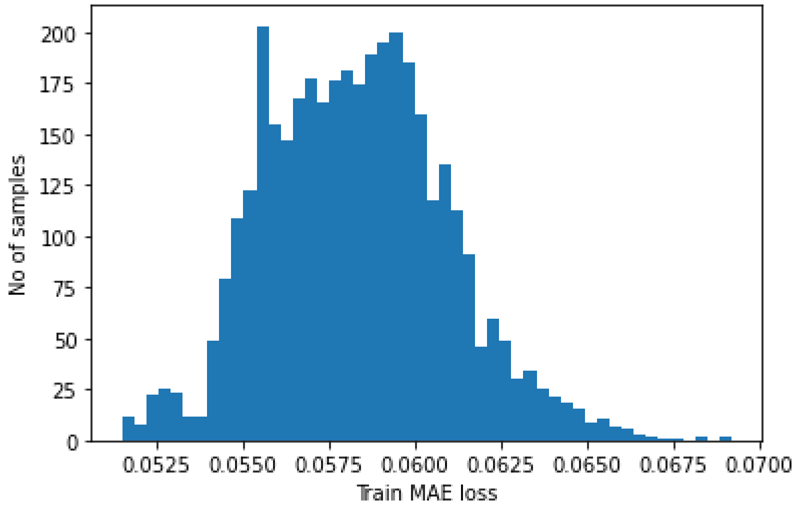
Detecting anomalies

We will detect anomalies by determining how well our model can reconstruct the input data.

1. Find MAE loss on training samples.
2. Find max MAE loss value. This is the worst our model has performed trying to reconstruct a sample. We will make this the `threshold` for anomaly detection.
3. If the reconstruction loss for a sample is greater than this `threshold` value then we can infer that the model is seeing a pattern that it isn't familiar with. We will label this sample as an `anomaly`.

```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
```

```
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```

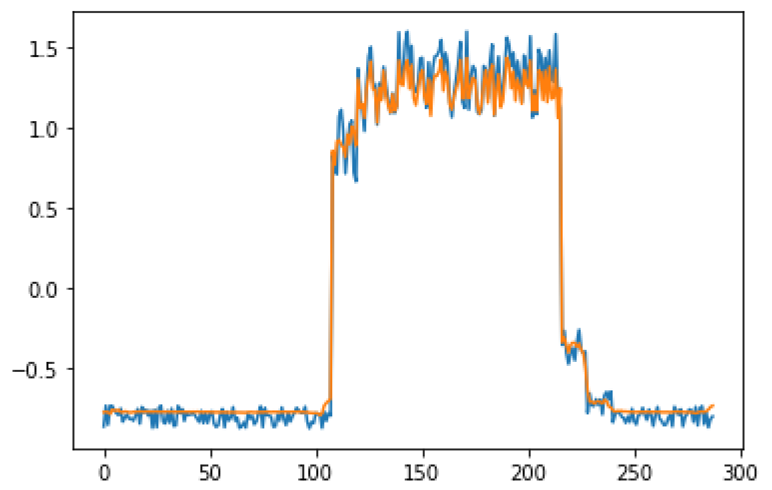



```
Reconstruction error threshold:  0.06919796281558628
```

Compare reconstruction

Just for fun, let's see how our model has recontructed the first sample. This is the 288 timesteps from day 1 of our training dataset.

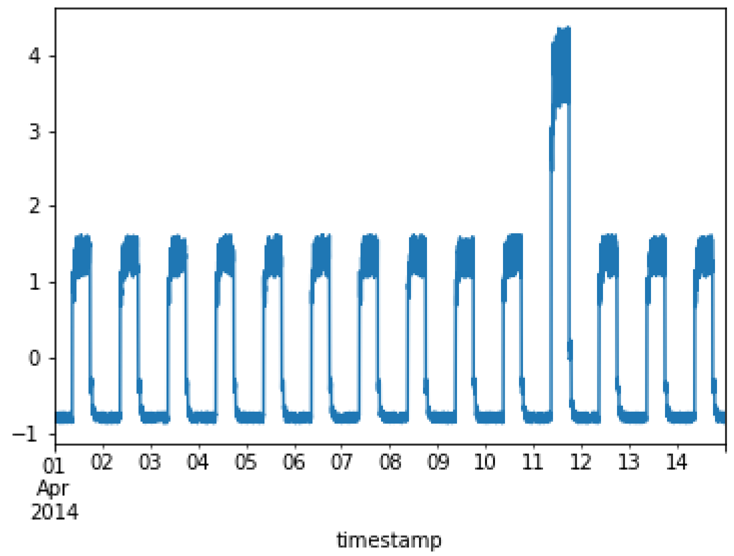
```
# Checking how the first sequence is learnt
plt.plot(x_train[0])
plt.plot(x_train_pred[0])
plt.show()
```



Prepare test data

```
df_test_value = (df_daily_jumpsup - training_mean) / training_std
fig, ax = plt.subplots()
df_test_value.plot(legend=False, ax=ax)
plt.show()
```

```
# Create sequences from test values.
x_test = create_sequences(df_test_value.values)
print("Test input shape: ", x_test.shape)

# Get test MAE loss.
x_test_pred = model.predict(x_test)
test_mae_loss = np.mean(np.abs(x_test_pred - x_test), axis=1)
test_mae_loss = test_mae_loss.reshape((-1))

plt.hist(test_mae_loss, bins=50)
plt.xlabel("test MAE loss")
plt.ylabel("No of samples")
plt.show()

# Detect all the samples which are anomalies.
anomalies = test_mae_loss > threshold
print("Number of anomaly samples: ", np.sum(anomalies))
print("Indices of anomaly samples: ", np.where(anomalies))
```




```
Test input shape:  (3745, 288, 1)
```

Plot anomalies

We now know the samples of the data which are anomalies. With this, we will find the corresponding `timestamps` from the original test data. We will be using the following method to do that:

Let's say time_steps = 3 and we have 10 training values. Our `x_train` will look like this:

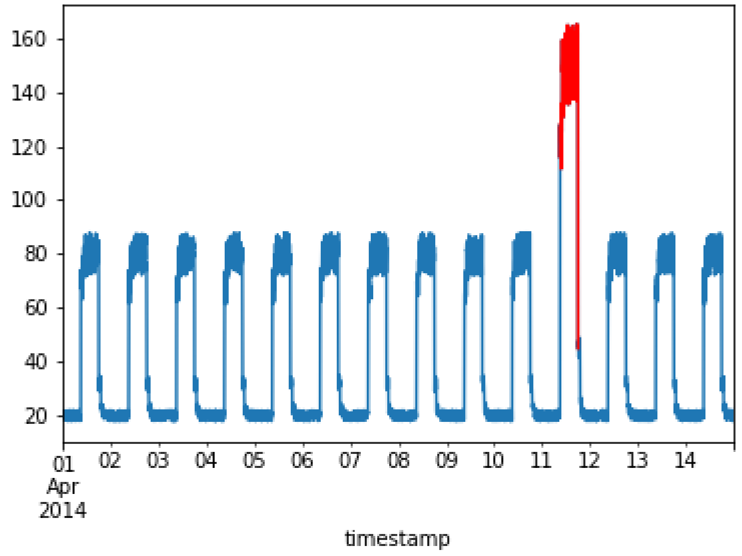
- 0, 1, 2
- 1, 2, 3
- 2, 3, 4
- 3, 4, 5
- 4, 5, 6
- 5, 6, 7
- 6, 7, 8
- 7, 8, 9

All except the initial and the final time_steps-1 data values, will appear in `time_steps` number of samples. So, if we know that the samples [(3, 4, 5), (4, 5, 6), (5, 6, 7)] are anomalies, we can say that the data point 5 is an anomaly.

```
            2891. 2892. 2893. 2894. 2895. 2896. 2897. 2898. 2899. 2900. 2901.
# data i is an anomaly if samples [(i - timesteps + 1) to (i)] are anomalies
anomalous_data_indices = []
for data_idx in range(TIME_STEPS - 1, len(df_test_value) - TIME_STEPS + 1):
    if np.all(anomalies[data_idx - TIME_STEPS + 1 : data_idx]):
        anomalous_data_indices.append(data_idx)
            2968, 2969, 2970, 2971, 2972, 2973, 2974, 2975, 2976, 2977, 2978,
```

Let's overlay the anomalies on the original test data plot.

```
df_subset = df_daily_jumpsup.iloc[anomalous_data_indices]
fig, ax = plt.subplots()
df_daily_jumpsup.plot(legend=False, ax=ax)
df_subset.plot(legend=False, ax=ax, color="r")
plt.show()
```



Exercises:

In this problem you are to compare different ideas of stochastic optimization. Ensure, that the algorithms are compared in the same setting: same initialization (fix the seed!) and same amount of epochs.

Learning rate schedule

- Train model using SGD optimizer with default hyperparameters.
- Train model using SGD optimizer with learning rate decay. Instructions can be found via this link.
- Compare the results.

  Optional: Log results with wandb

==YOUR ANSWER==

Train model using SGD optimizer with default hyperparameters.:

```
# Normalize and save the mean and std we get,
# for normalizing test data.
training_mean = df_small_noise.mean()
training_std = df_small_noise.std()
df_training_value = (df_small_noise - training_mean) / training_std
print("Number of training samples:", len(df_training_value))
```

```
Number of training samples: 4032
```

```python
TIME_STEPS = 288

# Generated training sequences for use in the model.
def create_sequences(values, time_steps=TIME_STEPS):
    output = []
    for i in range(len(values) - time_steps + 1):
        output.append(values[i : (i + time_steps)])
    return np.stack(output)


x_train = create_sequences(df_training_value.values)
print("Training input shape: ", x_train.shape)
```

```
Training input shape:  (3745, 288, 1)
```

```python
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01), loss="mse")
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv1d_16 (Conv1D)          (None, 144, 32)           256       
                                                                 
 dropout_16 (Dropout)        (None, 144, 32)           0         
                                                                 
 conv1d_17 (Conv1D)          (None, 72, 16)            3600      
                                                                 
 conv1d_transpose_24 (Conv1D  (None, 144, 16)          1808      
 Transpose)                                                      
                                                                 
 dropout_17 (Dropout)        (None, 144, 16)           0         
                                                                 
 conv1d_transpose_25 (Conv1D  (None, 288, 32)          3616      
 Transpose)                                                      
```

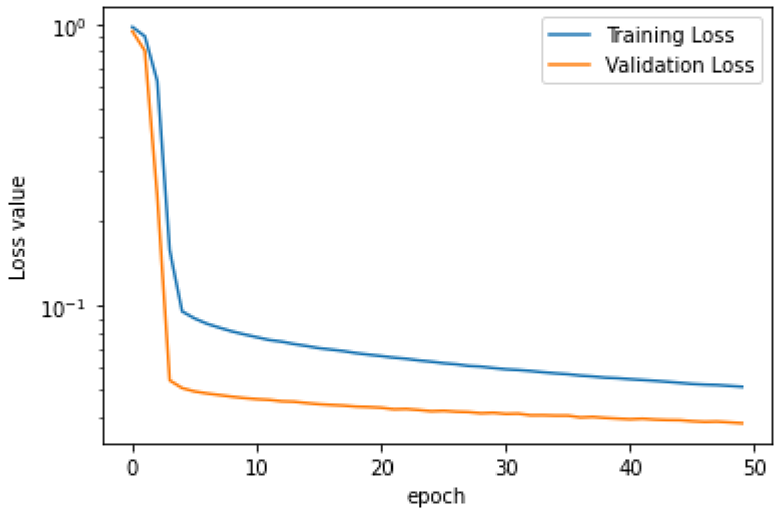
```
 conv1d_transpose_26 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

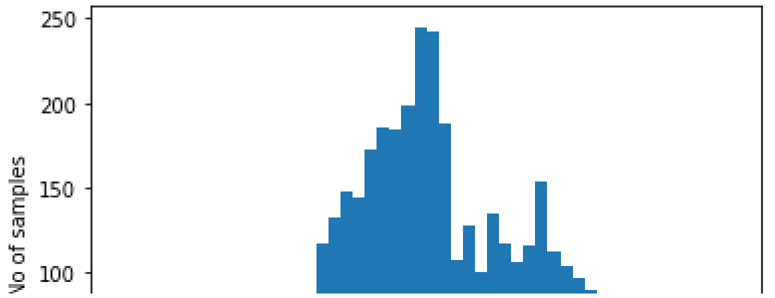
```
27/27 [==============================] - 2s 82ms/step - loss: 0.0706 - val_loss: 0.0446
Epoch 17/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0697 - val_loss: 0.0443
Epoch 18/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0689 - val_loss: 0.0441
Epoch 19/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0678 - val_loss: 0.0437
Epoch 20/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0669 - val_loss: 0.0436
Epoch 21/50

27/27 [==============================] - 2s 81ms/step - loss: 0.0662 - val_loss: 0.0433
Epoch 22/50
27/27 [==============================] - 3s 101ms/step - loss: 0.0653 - val_loss: 0.0428
Epoch 23/50
27/27 [==============================] - 3s 124ms/step - loss: 0.0647 - val_loss: 0.0429
Epoch 24/50
27/27 [==============================] - 4s 133ms/step - loss: 0.0638 - val_loss: 0.0425
Epoch 25/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0632 - val_loss: 0.0421
Epoch 26/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0624 - val_loss: 0.0422
Epoch 27/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0618 - val_loss: 0.0419
Epoch 28/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0610 - val_loss: 0.0418
Epoch 29/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0607 - val_loss: 0.0414
Epoch 30/50
27/27 [==============================] - 2s 78ms/step - loss: 0.0600 - val_loss: 0.0416
Epoch 31/50
27/27 [==============================] - 2s 78ms/step - loss: 0.0594 - val_loss: 0.0412
Epoch 32/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0590 - val_loss: 0.0413
Epoch 33/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0585 - val_loss: 0.0407
Epoch 34/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0579 - val_loss: 0.0407
Epoch 35/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0574 - val_loss: 0.0406
Epoch 36/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0570 - val_loss: 0.0406
Epoch 37/50
```

```
27/27 [==============================] - 2s 79ms/step - loss: 0.0564 - val_loss: 0.0400
Epoch 38/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0560 - val_loss: 0.0402
Epoch 39/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0554 - val_loss: 0.0398
Epoch 40/50
27/27 [==============================] - 2s 78ms/step - loss: 0.0552 - val_loss: 0.0396
Epoch 41/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0547 - val_loss: 0.0394
Epoch 42/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0544 - val_loss: 0.0396
Epoch 43/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0540 - val_loss: 0.0393
Epoch 44/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0536 - val_loss: 0.0392
Epoch 45/50
```

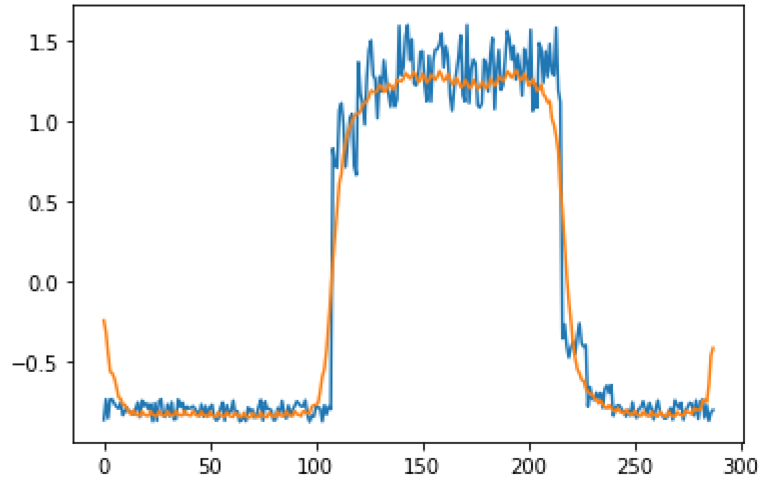
```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```



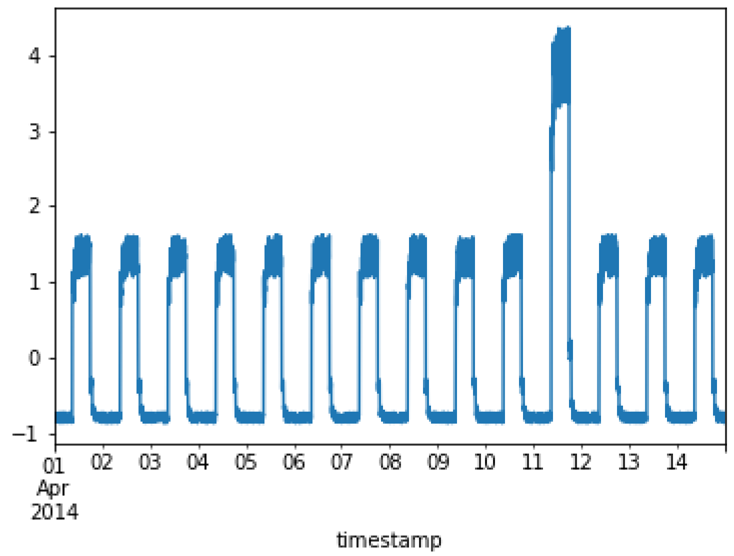

```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```



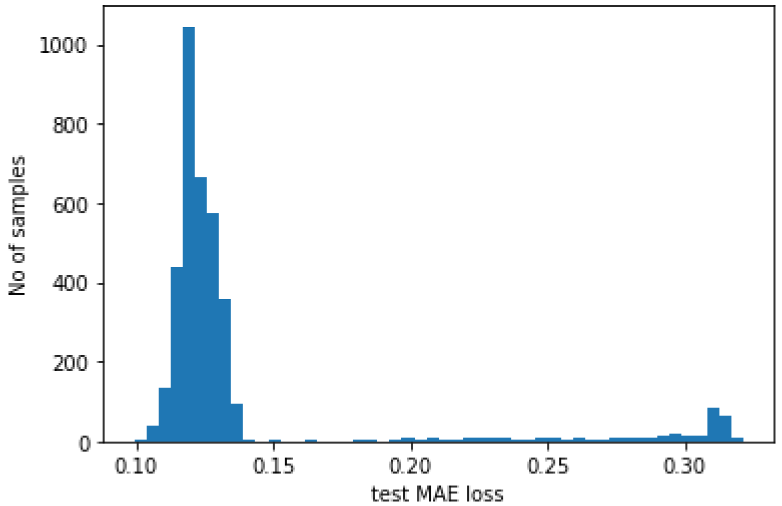

```
# Checking how the first sequence is learnt
plt.plot(x_train[0])
plt.plot(x_train_pred[0])
plt.show()
```




```
df_test_value = (df_daily_jumpsup - training_mean) / training_std
fig, ax = plt.subplots()
df_test_value.plot(legend=False, ax=ax)
plt.show()

# Create sequences from test values.
x_test = create_sequences(df_test_value.values)
print("Test input shape: ", x_test.shape)

# Get test MAE loss.
x_test_pred = model.predict(x_test)
test_mae_loss = np.mean(np.abs(x_test_pred - x_test), axis=1)
test_mae_loss = test_mae_loss.reshape((-1))

plt.hist(test_mae_loss, bins=50)
plt.xlabel("test MAE loss")
plt.ylabel("No of samples")
plt.show()

# Detect all the samples which are anomalies.
anomalies = test_mae_loss > threshold
print("Number of anomaly samples: ", np.sum(anomalies))
print("Indices of anomaly samples: ", np.where(anomalies))
```




```
Test input shape:  (3745, 288, 1)
```




```
Number of anomaly samples:  390
Indices of anomaly samples:  (array([2704, 2705, 2706, 2707, 2708, 2709, 2710, 2711, 2712, 2713, 2714,
       2715, 2716, 2717, 2718, 2719, 2720, 2721, 2722, 2723, 2724, 2725,
       2726, 2727, 2728, 2729, 2730, 2731, 2732, 2733, 2734, 2735, 2736,
       2737, 2738, 2739, 2740, 2741, 2742, 2743, 2744, 2745, 2746, 2747,
       2748, 2749, 2750, 2751, 2752, 2753, 2754, 2755, 2756, 2757, 2758,
       2759, 2760, 2761, 2762, 2763, 2764, 2765, 2766, 2767, 2768, 2769,
       2770, 2771, 2772, 2773, 2774, 2775, 2776, 2777, 2778, 2779, 2780,
       2781, 2782, 2783, 2784, 2785, 2786, 2787, 2788, 2789, 2790, 2791,
       2792, 2793, 2794, 2795, 2796, 2797, 2798, 2799, 2800, 2801, 2802,
       2803, 2804, 2805, 2806, 2807, 2808, 2809, 2810, 2811, 2812, 2813,
       2814, 2815, 2816, 2817, 2818, 2819, 2820, 2821, 2822, 2823, 2824,
       2825, 2826, 2827, 2828, 2829, 2830, 2831, 2832, 2833, 2834, 2835,
       2836, 2837, 2838, 2839, 2840, 2841, 2842, 2843, 2844, 2845, 2846,
       2847, 2848, 2849, 2850, 2851, 2852, 2853, 2854, 2855, 2856, 2857,
       2858, 2859, 2860, 2861, 2862, 2863, 2864, 2865, 2866, 2867, 2868,
       2869, 2870, 2871, 2872, 2873, 2874, 2875, 2876, 2877, 2878, 2879,
       2880, 2881, 2882, 2883, 2884, 2885, 2886, 2887, 2888, 2889, 2890,
       2891, 2892, 2893, 2894, 2895, 2896, 2897, 2898, 2899, 2900, 2901,
       2902, 2903, 2904, 2905, 2906, 2907, 2908, 2909, 2910, 2911, 2912,
       2913, 2914, 2915, 2916, 2917, 2918, 2919, 2920, 2921, 2922, 2923,
       2924, 2925, 2926, 2927, 2928, 2929, 2930, 2931, 2932, 2933, 2934,
       2935, 2936, 2937, 2938, 2939, 2940, 2941, 2942, 2943, 2944, 2945,
       2946, 2947, 2948, 2949, 2950, 2951, 2952, 2953, 2954, 2955, 2956,
       2957, 2958, 2959, 2960, 2961, 2962, 2963, 2964, 2965, 2966, 2967,
       2968, 2969, 2970, 2971, 2972, 2973, 2974, 2975, 2976, 2977, 2978,
       2979, 2980, 2981, 2982, 2983, 2984, 2985, 2986, 2987, 2988, 2989,
       2990, 2991, 2992, 2993, 2994, 2995, 2996, 2997, 2998, 2999, 3000,
       3001, 3002, 3003, 3004, 3005, 3006, 3007, 3008, 3009, 3010, 3011,
       3012, 3013, 3014, 3015, 3016, 3017, 3018, 3019, 3020, 3021, 3022,
       3023, 3024, 3025, 3026, 3027, 3028, 3029, 3030, 3031, 3032, 3033,
       3034, 3035, 3036, 3037, 3038, 3039, 3040, 3041, 3042, 3043, 3044,
```

Train model using SGD optimizer with learning rate decay. Instructions can be found via this link.

```
            3067, 3068, 3069, 3070, 3071, 3072, 3073, 3074, 3075, 3076, 3077,
```

```
import tensorflow as tf
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
initial_learning_rate = 0.01
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate,
    decay_steps=100000,
    decay_rate=0.7,
    staircase=True)
"""
initial_learning_rate = 0.1
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate,
    decay_steps=100000,
    decay_rate=0.96,
    staircase=True)

model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=lr_schedule),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
              """
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=lr_schedule), loss="mse")
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_18 (Conv1D)          (None, 144, 32)           256

 dropout_18 (Dropout)        (None, 144, 32)           0

 conv1d_19 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_27 (Conv1D  (None, 144, 16)          1808
 Transpose)
```

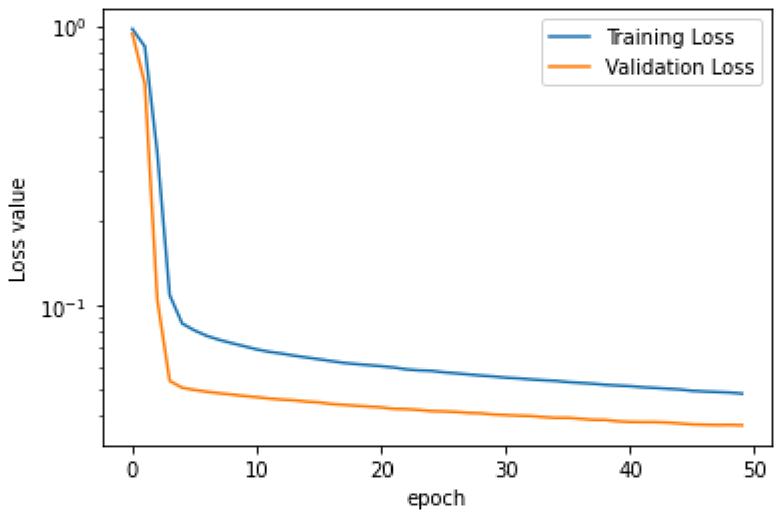
```
 dropout_19 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_28 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_29 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

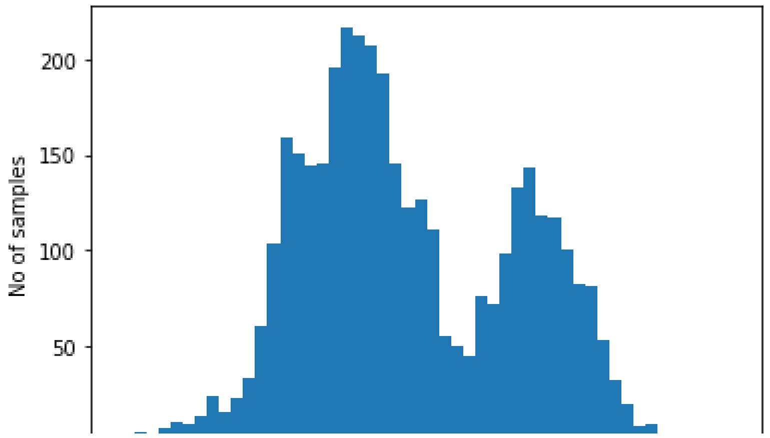
```
27/27 [==============================] - 2s 78ms/step - loss: 0.0639 - val_loss: 0.0447
Epoch 17/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0629 - val_loss: 0.0441
Epoch 18/50
27/27 [==============================] - 2s 78ms/step - loss: 0.0620 - val_loss: 0.0438
Epoch 19/50

27/27 [==============================] - 2s 79ms/step - loss: 0.0614 - val_loss: 0.0435
Epoch 20/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0609 - val_loss: 0.0432
Epoch 21/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0603 - val_loss: 0.0429
Epoch 22/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0597 - val_loss: 0.0424
Epoch 23/50
27/27 [==============================] - 2s 78ms/step - loss: 0.0588 - val_loss: 0.0423
Epoch 24/50
27/27 [==============================] - 2s 78ms/step - loss: 0.0583 - val_loss: 0.0420
Epoch 25/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0581 - val_loss: 0.0416
Epoch 26/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0573 - val_loss: 0.0415
Epoch 27/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0569 - val_loss: 0.0413
Epoch 28/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0563 - val_loss: 0.0410
Epoch 29/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0558 - val_loss: 0.0409
Epoch 30/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0554 - val_loss: 0.0405
Epoch 31/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0549 - val_loss: 0.0403
Epoch 32/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0545 - val_loss: 0.0401
27/27 [==============================] - 2s 78ms/step - loss: 0.0541 - val_loss: 0.0400
Epoch 33/50
Epoch 34/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0537 - val_loss: 0.0396
Epoch 35/50
```

```
27/27 [==============================] - 2s 79ms/step - loss: 0.0534 - val_loss: 0.0394
Epoch 36/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0529 - val_loss: 0.0394
Epoch 37/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0526 - val_loss: 0.0391
Epoch 38/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0522 - val_loss: 0.0387
Epoch 39/50
27/27 [==============================] - 2s 78ms/step - loss: 0.0517 - val_loss: 0.0387
Epoch 40/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0514 - val_loss: 0.0383
Epoch 41/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0511 - val_loss: 0.0381
Epoch 42/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0507 - val_loss: 0.0380
Epoch 43/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0504 - val_loss: 0.0380
Epoch 44/50
27/27 [==============================] - 2s 78ms/step - loss: 0.0501 - val_loss: 0.0379
Epoch 45/50
```

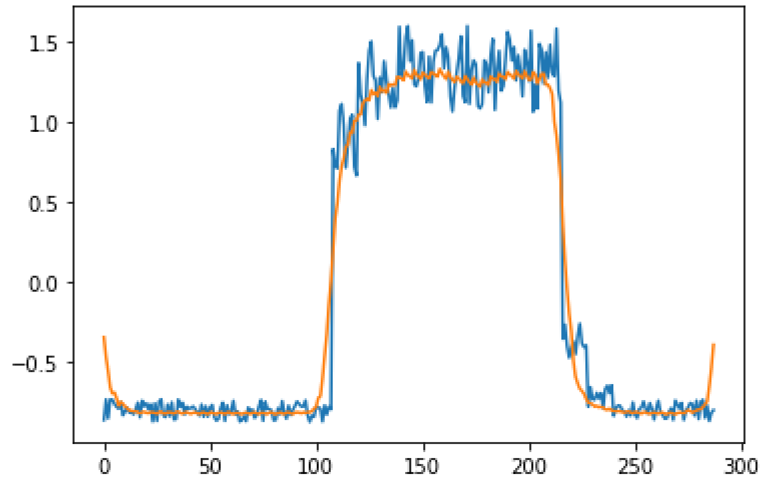
```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```




```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
# Checking how the first sequence is learnt
plt.plot(x_train[0])
plt.plot(x_train_pred[0])
plt.show()
```



ДОБАВЛЕНИЕ LEARNING RATE ускоряет уменьшение validation loss до предельной точности, то есть нужно меньшее количество шагов до достижения нужной ошибки. На графике выглядит, как сдвиг на константу вниз + выше крутизна "падения графика"

Acceleration

- Train model using SGD optimizer with default hyperparameters.
- Train model using SGD optimizer with momentum term.
- Train model using SGD optimizer with nesterov momentum term. Instructions can be found via this link.
- Compare the results.

  Optional: Log results with wandb

*Train model using SGD optimizer with default hyperparameters. *

==YOUR ANSWER==

```
# Normalize and save the mean and std we get,
# for normalizing test data.
training_mean = df_small_noise.mean()
training_std = df_small_noise.std()
df_training_value = (df_small_noise - training_mean) / training_std
print("Number of training samples:", len(df_training_value))
```

```
Number of training samples: 4032
```

```
TIME_STEPS = 288

# Generated training sequences for use in the model.
def create_sequences(values, time_steps=TIME_STEPS):
    output = []
    for i in range(len(values) - time_steps + 1):
        output.append(values[i : (i + time_steps)])
    return np.stack(output)


x_train = create_sequences(df_training_value.values)
print("Training input shape: ", x_train.shape)
```

```
Training input shape:  (3745, 288, 1)
```

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01), loss="mse")
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_20 (Conv1D)          (None, 144, 32)           256

 dropout_20 (Dropout)        (None, 144, 32)           0

 conv1d_21 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_30 (Conv1D  (None, 144, 16)          1808
```

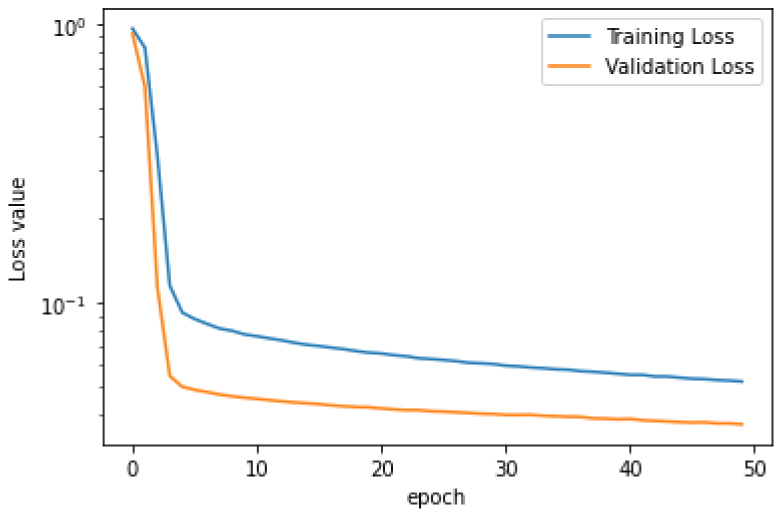
```
 Transpose)

 dropout_21 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_31 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_32 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

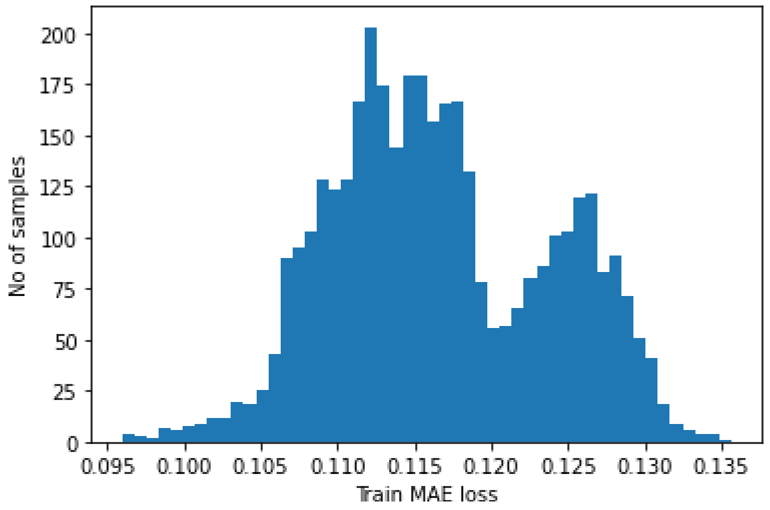
```
27/27 [==============================] - 2s 86ms/step - loss: 0.0704 - val_loss: 0.0436
Epoch 17/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0694 - val_loss: 0.0431
Epoch 18/50

27/27 [==============================] - 2s 85ms/step - loss: 0.0686 - val_loss: 0.0428
Epoch 19/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0675 - val_loss: 0.0425
Epoch 20/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0666 - val_loss: 0.0424
Epoch 21/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0662 - val_loss: 0.0420
Epoch 22/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0653 - val_loss: 0.0417
Epoch 23/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0647 - val_loss: 0.0415
Epoch 24/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0636 - val_loss: 0.0414
Epoch 25/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0632 - val_loss: 0.0411
Epoch 26/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0627 - val_loss: 0.0409
Epoch 27/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0621 - val_loss: 0.0407
Epoch 28/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0613 - val_loss: 0.0405
Epoch 29/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0610 - val_loss: 0.0403
Epoch 30/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0607 - val_loss: 0.0402
Epoch 31/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0598 - val_loss: 0.0399
Epoch 32/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0595 - val_loss: 0.0398
Epoch 33/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0590 - val_loss: 0.0399
Epoch 34/50
```

```
27/27 [==============================] - 2s 84ms/step - loss: 0.0586 - val_loss: 0.0396
Epoch 35/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0581 - val_loss: 0.0394
Epoch 36/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0579 - val_loss: 0.0393
Epoch 37/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0572 - val_loss: 0.0392
Epoch 38/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0569 - val_loss: 0.0387
Epoch 39/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0565 - val_loss: 0.0386
Epoch 40/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0560 - val_loss: 0.0385
Epoch 41/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0555 - val_loss: 0.0386
Epoch 42/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0555 - val_loss: 0.0381
Epoch 43/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0548 - val_loss: 0.0379
Epoch 44/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0547 - val_loss: 0.0378
Epoch 45/50
```

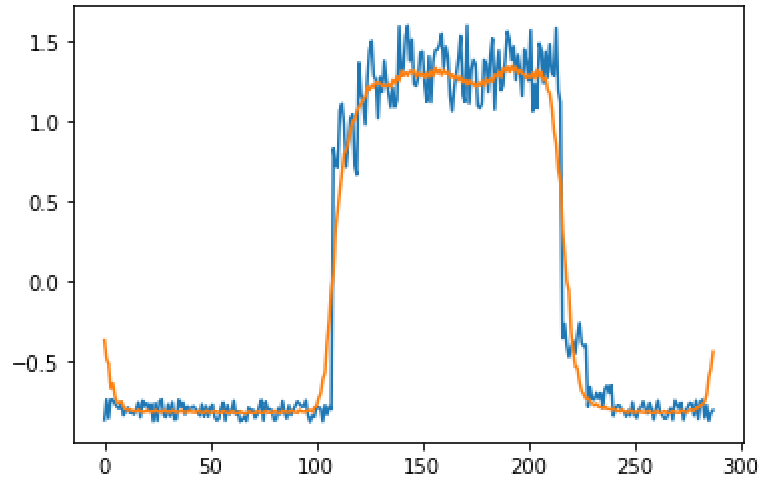
```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```




```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
# Checking how the first sequence is learnt
plt.plot(x_train[0])
plt.plot(x_train_pred[0])
plt.show()
```



ABUSING MOMENTUM

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
```

```
)
#opt = tf.keras.optimizers.SGD(learning_rate=0.1, momentum=0.9)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01, momentum=0.9), loss="mse")
model.summary()
```

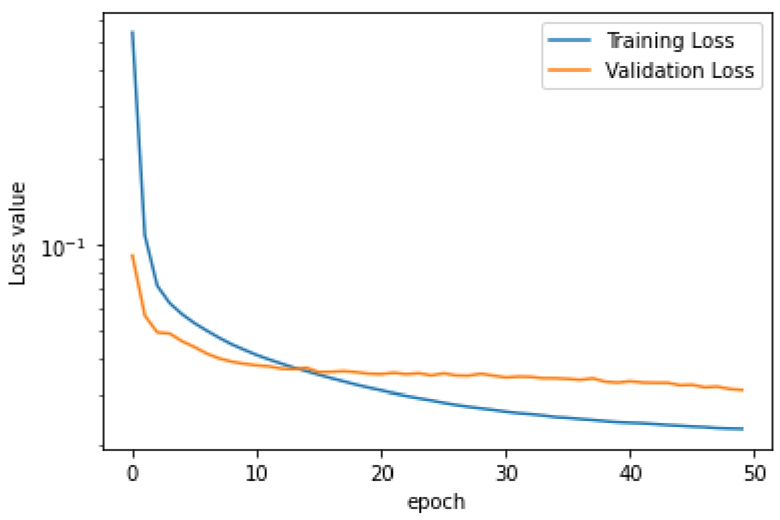
```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_22 (Conv1D)          (None, 144, 32)           256

 dropout_22 (Dropout)        (None, 144, 32)           0

 conv1d_23 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_33 (Conv1D  (None, 144, 16)          1808
 Transpose)

 dropout_23 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_34 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_35 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

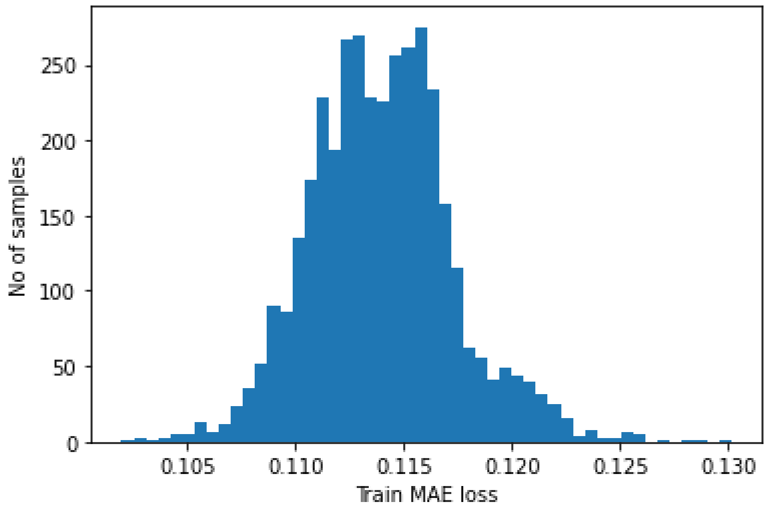
```
Epoch 16/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0352 - val_loss: 0.0360
Epoch 17/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0343 - val_loss: 0.0361
Epoch 18/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0334 - val_loss: 0.0363
Epoch 19/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0326 - val_loss: 0.0359
Epoch 20/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0318 - val_loss: 0.0355
Epoch 21/50
27/27 [==============================] - 3s 107ms/step - loss: 0.0311 - val_loss: 0.0354
Epoch 22/50
27/27 [==============================] - 3s 103ms/step - loss: 0.0304 - val_loss: 0.0358
Epoch 23/50
27/27 [==============================] - 2s 87ms/step - loss: 0.0298 - val_loss: 0.0354
27/27 [==============================] - 2s 86ms/step - loss: 0.0293 - val_loss: 0.0357
Epoch 24/50
Epoch 25/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0288 - val_loss: 0.0350
```

```
Epoch 26/50
27/27 [==============================] - 2s 87ms/step - loss: 0.0282 - val_loss: 0.0356
Epoch 27/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0278 - val_loss: 0.0350
Epoch 28/50
27/27 [==============================] - 2s 87ms/step - loss: 0.0273 - val_loss: 0.0350
Epoch 29/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0270 - val_loss: 0.0355
Epoch 30/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0267 - val_loss: 0.0350
Epoch 31/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0263 - val_loss: 0.0345
Epoch 32/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0260 - val_loss: 0.0347
Epoch 33/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0258 - val_loss: 0.0346
Epoch 34/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0255 - val_loss: 0.0343
Epoch 35/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0252 - val_loss: 0.0342
Epoch 36/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0250 - val_loss: 0.0341
Epoch 37/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0248 - val_loss: 0.0338
Epoch 38/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0246 - val_loss: 0.0342
Epoch 39/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0244 - val_loss: 0.0333
Epoch 40/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0242 - val_loss: 0.0331
Epoch 41/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0241 - val_loss: 0.0334

Epoch 42/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0240 - val_loss: 0.0331
Epoch 43/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0238 - val_loss: 0.0330
Epoch 44/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0236 - val_loss: 0.0330
```

```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```

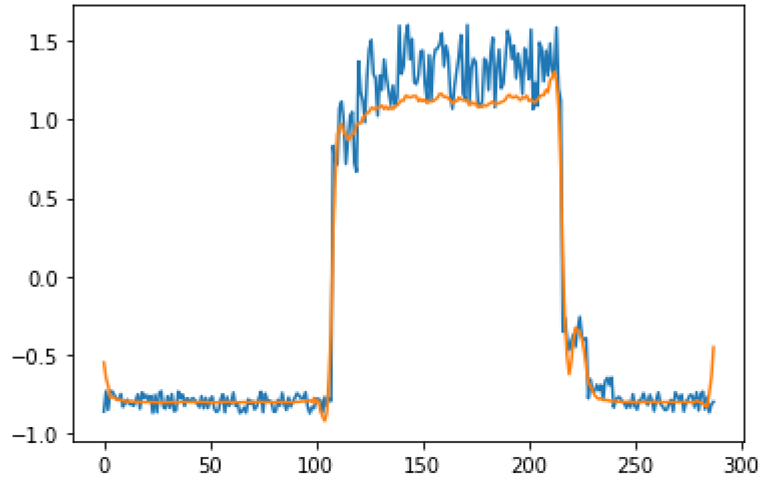


```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
Reconstruction error threshold:  0.13013304441452291
```

```
# Checking how the first sequence is learnt
plt.plot(x_train[0])
plt.plot(x_train_pred[0])
plt.show()
```



Train model using SGD optimizer with nesterov momentum term. Instructions can be found via this link.

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
```

```
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01, momentum  = 0.8, nesterov=True), loss="mse")
model.summary()
```

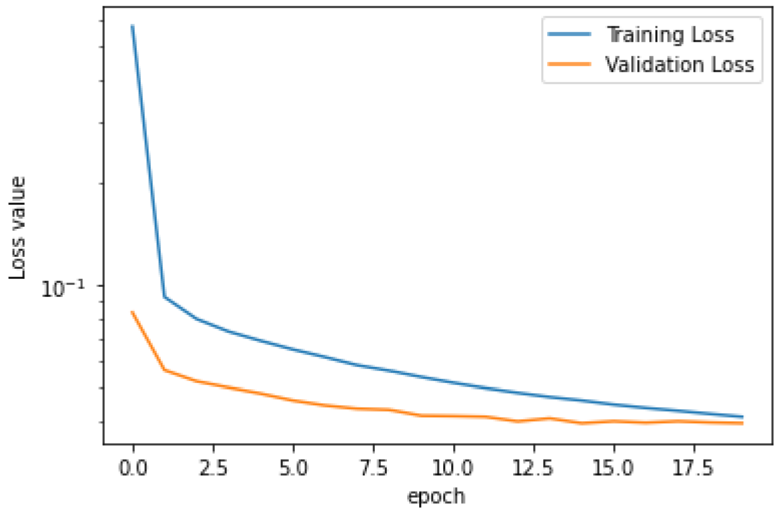
```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv1d_12 (Conv1D)          (None, 144, 32)           256       
                                                                 
 dropout_12 (Dropout)        (None, 144, 32)           0         
                                                                 
 conv1d_13 (Conv1D)          (None, 72, 16)            3600      
                                                                 
 conv1d_transpose_18 (Conv1D  (None, 144, 16)          1808      
 Transpose)                                                      
                                                                 
 dropout_13 (Dropout)        (None, 144, 16)           0         
                                                                 
 conv1d_transpose_19 (Conv1D  (None, 288, 32)          3616      
 Transpose)                                                      
                                                                 
 conv1d_transpose_20 (Conv1D  (None, 288, 1)           225       
 Transpose)                                                      
                                                                 
=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

```
Epoch 1/50
27/27 [==============================] - 5s 144ms/step - loss: 0.5740 - val_loss: 0.0832
Epoch 2/50
27/27 [==============================] - 3s 122ms/step - loss: 0.0924 - val_loss: 0.0563
Epoch 3/50
```

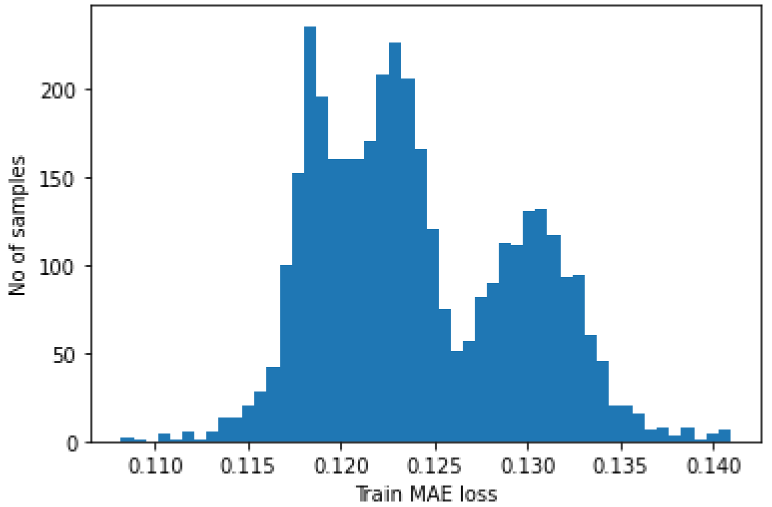
```
27/27 [==============================] - 4s 147ms/step - loss: 0.0796 - val_loss: 0.0523
Epoch 4/50
27/27 [==============================] - 3s 129ms/step - loss: 0.0731 - val_loss: 0.0501
Epoch 5/50
27/27 [==============================] - 3s 126ms/step - loss: 0.0687 - val_loss: 0.0482
Epoch 6/50
27/27 [==============================] - 3s 128ms/step - loss: 0.0648 - val_loss: 0.0460
Epoch 7/50
27/27 [==============================] - 4s 131ms/step - loss: 0.0616 - val_loss: 0.0445
Epoch 8/50
27/27 [==============================] - 3s 126ms/step - loss: 0.0583 - val_loss: 0.0435
Epoch 9/50
27/27 [==============================] - 3s 128ms/step - loss: 0.0561 - val_loss: 0.0432
Epoch 10/50
27/27 [==============================] - 3s 129ms/step - loss: 0.0538 - val_loss: 0.0415
Epoch 11/50
27/27 [==============================] - 3s 126ms/step - loss: 0.0517 - val_loss: 0.0414
Epoch 12/50
27/27 [==============================] - 3s 124ms/step - loss: 0.0499 - val_loss: 0.0412
Epoch 13/50
27/27 [==============================] - 3s 126ms/step - loss: 0.0484 - val_loss: 0.0399
Epoch 14/50
27/27 [==============================] - 3s 126ms/step - loss: 0.0471 - val_loss: 0.0408
Epoch 15/50
27/27 [==============================] - 4s 134ms/step - loss: 0.0460 - val_loss: 0.0394
Epoch 16/50
27/27 [==============================] - 3s 128ms/step - loss: 0.0447 - val_loss: 0.0400
Epoch 17/50
27/27 [==============================] - 4s 133ms/step - loss: 0.0437 - val_loss: 0.0396
Epoch 18/50
27/27 [==============================] - 3s 129ms/step - loss: 0.0429 - val_loss: 0.0400
Epoch 19/50
27/27 [==============================] - 3s 129ms/step - loss: 0.0420 - val_loss: 0.0396
Epoch 20/50
27/27 [==============================] - 3s 130ms/step - loss: 0.0412 - val_loss: 0.0395
```

```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```



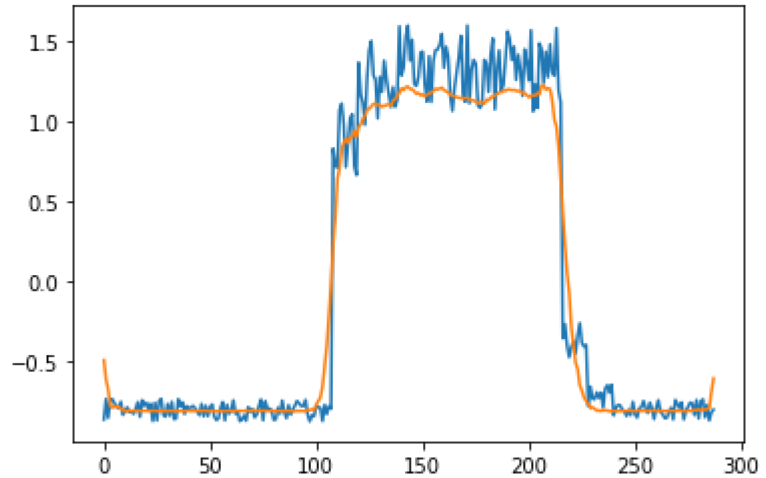

```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
```

```
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
Reconstruction error threshold:  0.1409303680635231
```

```
# Checking how the first sequence is learnt
plt.plot(x_train[0])
plt.plot(x_train_pred[0])
plt.show()
```



ВЫВОД:

Заявим, что training loss в целом не изменяется, хотя с nesterov momentum он падает чуть быстрее, однако затем график переламывается и он идёт как остальные. Сравнивая momentum с чистым SGD видно, что скорость и финальная точность выше у чистого, возможно, из-за неверно подобранного момента, так как он может быть слишком низким. Тем не менее, nesterov momentum показал лучший результат, дав самую высокую точность в конце и самую маленькую validation loss в начале.

Adaptive methods

- Train model using SGD optimizer with default hyperparameters.
- Train model using any adaptive method. Instructions can be found via [this]() link.
- Compare the results.
- Try to perform different runs of SGD + Momentum and select the best hyperparameters. Do the same for the Adam in the similar setting. Compare the results.

  Optional: Log results with wandb

==YOUR ANSWER==

Train model using SGD optimizer with default hyperparameters.

```
# Normalize and save the mean and std we get,
# for normalizing test data.
training_mean = df_small_noise.mean()
training_std = df_small_noise.std()
df_training_value = (df_small_noise - training_mean) / training_std
print("Number of training samples:", len(df_training_value))
```

```
Number of training samples: 4032
```

```
TIME_STEPS = 288

# Generated training sequences for use in the model.
def create_sequences(values, time_steps=TIME_STEPS):
    output = []
    for i in range(len(values) - time_steps + 1):
        output.append(values[i : (i + time_steps)])
    return np.stack(output)


x_train = create_sequences(df_training_value.values)
print("Training input shape: ", x_train.shape)
```

```
Training input shape:  (3745, 288, 1)
```

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
```

```
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
#tf.keras.optimizers.Adam(learning_rate=0.1)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01), loss="mse")
model.summary()
```

```
Model: "sequential_30"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_60 (Conv1D)          (None, 144, 32)           256

 dropout_60 (Dropout)        (None, 144, 32)           0

 conv1d_61 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_90 (Conv1D  (None, 144, 16)          1808
 Transpose)

 dropout_61 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_91 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_92 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

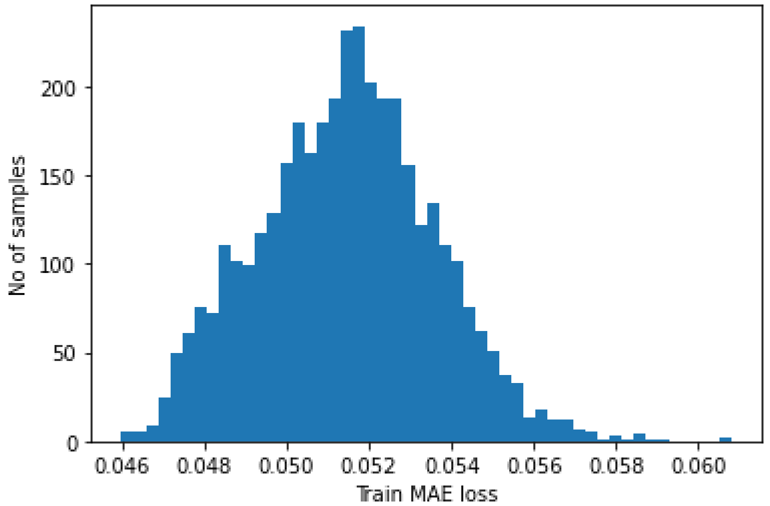
```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

```
27/27 [==============================] - 2s 88ms/step - loss: 0.0685 - val_loss: 0.0438
Epoch 17/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0674 - val_loss: 0.0433
Epoch 18/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0660 - val_loss: 0.0427
Epoch 19/50
27/27 [==============================] - 2s 90ms/step - loss: 0.0650 - val_loss: 0.0424
Epoch 20/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0641 - val_loss: 0.0421
Epoch 21/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0632 - val_loss: 0.0418
```

```
Epoch 22/50
27/27 [==============================] - 2s 88ms/step - loss: 0.0624 - val_loss: 0.0414
Epoch 23/50
27/27 [==============================] - 2s 91ms/step - loss: 0.0618 - val_loss: 0.0412
Epoch 24/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0608 - val_loss: 0.0408
Epoch 25/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0602 - val_loss: 0.0404
Epoch 26/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0595 - val_loss: 0.0402
Epoch 27/50
27/27 [==============================] - 2s 90ms/step - loss: 0.0592 - val_loss: 0.0401
Epoch 28/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0582 - val_loss: 0.0398
Epoch 29/50
27/27 [==============================] - 2s 90ms/step - loss: 0.0577 - val_loss: 0.0395
Epoch 30/50
27/27 [==============================] - 2s 88ms/step - loss: 0.0572 - val_loss: 0.0393
Epoch 31/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0568 - val_loss: 0.0392
Epoch 32/50
27/27 [==============================] - 2s 90ms/step - loss: 0.0564 - val_loss: 0.0390
Epoch 33/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0557 - val_loss: 0.0388
Epoch 34/50
27/27 [==============================] - 2s 88ms/step - loss: 0.0553 - val_loss: 0.0384
Epoch 35/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0548 - val_loss: 0.0385
Epoch 36/50
27/27 [==============================] - 2s 90ms/step - loss: 0.0542 - val_loss: 0.0382
Epoch 37/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0539 - val_loss: 0.0381

Epoch 38/50
27/27 [==============================] - 2s 90ms/step - loss: 0.0536 - val_loss: 0.0378
Epoch 39/50
27/27 [==============================] - 2s 88ms/step - loss: 0.0531 - val_loss: 0.0378
Epoch 40/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0528 - val_loss: 0.0376
Epoch 41/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0524 - val_loss: 0.0374
Epoch 42/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0521 - val_loss: 0.0373
Epoch 43/50
27/27 [==============================] - 2s 89ms/step - loss: 0.0517 - val_loss: 0.0371
Epoch 44/50
27/27 [==============================] - 2s 90ms/step - loss: 0.0513 - val_loss: 0.0370
Epoch 45/50
```

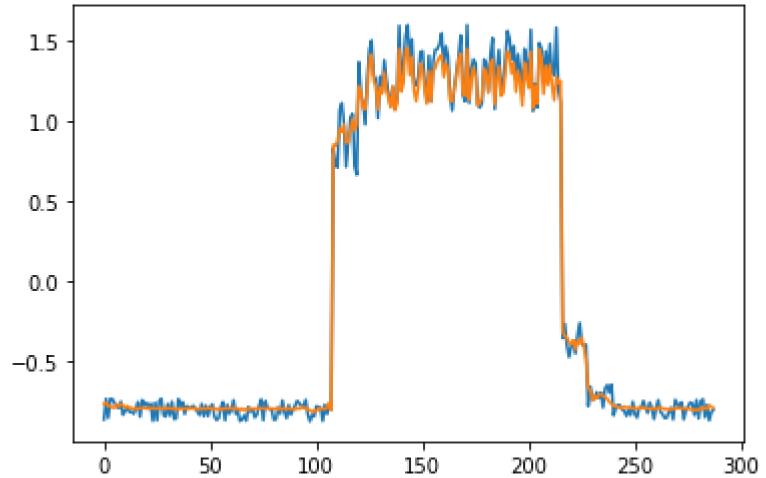
```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```


```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```

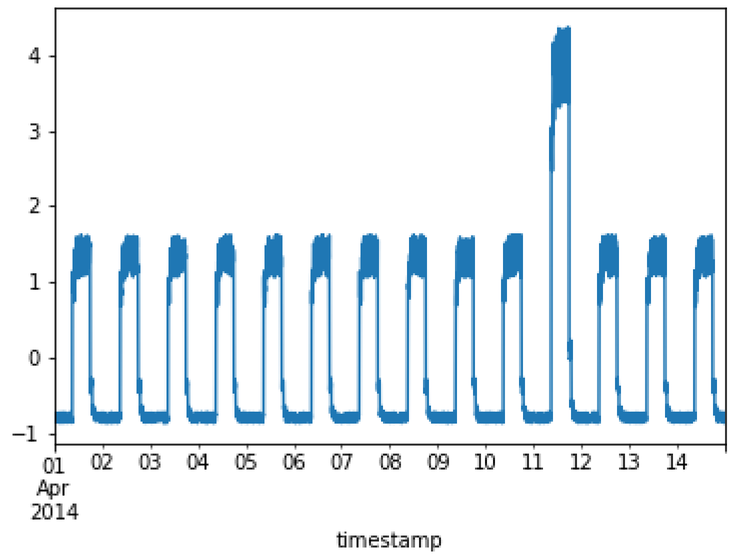



```
Reconstruction error threshold:  0.060795858691389974
```

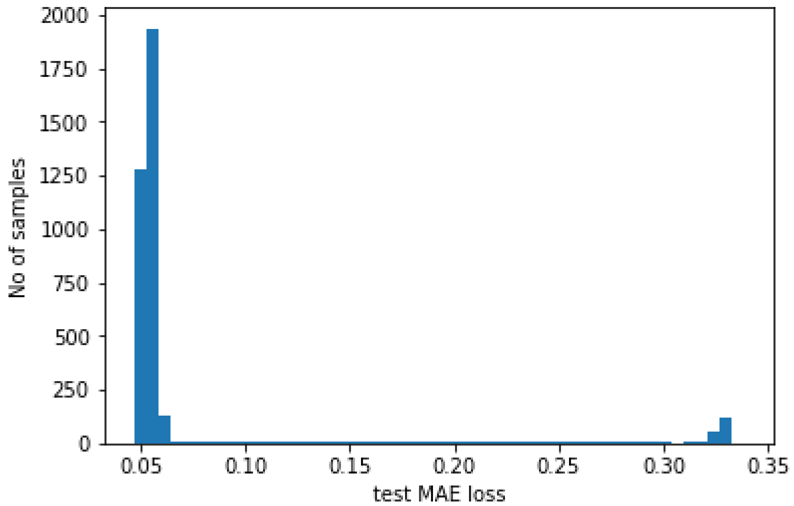
```
# Checking how the first sequence is learnt
plt.plot(x_train[0])
plt.plot(x_train_pred[0])
plt.show()
```



```
df_test_value = (df_daily_jumpsup - training_mean) / training_std
fig, ax = plt.subplots()
df_test_value.plot(legend=False, ax=ax)
plt.show()

# Create sequences from test values.
x_test = create_sequences(df_test_value.values)
print("Test input shape: ", x_test.shape)

# Get test MAE loss.
x_test_pred = model.predict(x_test)
test_mae_loss = np.mean(np.abs(x_test_pred - x_test), axis=1)
test_mae_loss = test_mae_loss.reshape((-1))

plt.hist(test_mae_loss, bins=50)
plt.xlabel("test MAE loss")
plt.ylabel("No of samples")
plt.show()

# Detect all the samples which are anomalies.
anomalies = test_mae_loss > threshold
print("Number of anomaly samples: ", np.sum(anomalies))
print("Indices of anomaly samples: ", np.where(anomalies))
```




```
Test input shape:  (3745, 288, 1)
```




```
Number of anomaly samples:  412
```

```
        2710, 2711, 2712, 2713, 2714, 2715, 2716, 2717, 2718, 2719, 2720,
```

Train model using any adaptive method. Instructions can be found via this link.

```
        2765, 2766, 2767, 2768, 2769, 2770, 2771, 2772, 2773, 2774, 2775,
```

```
        2798, 2799, 2800, 2801, 2802, 2803, 2804, 2805, 2806, 2807, 2808,
```

```
# Normalize and save the mean and std we get,
# for normalizing test data.
training_mean = df_small_noise.mean()
training_std = df_small_noise.std()
df_training_value = (df_small_noise - training_mean) / training_std
print("Number of training samples:", len(df_training_value))
```

```
Number of training samples: 4032
        2919, 2920, 2921, 2922, 2923, 2924, 2925, 2926, 2927, 2928, 2929,
```

```
TIME_STEPS = 288

# Generated training sequences for use in the model.
def create_sequences(values, time_steps=TIME_STEPS):
    output = []
    for i in range(len(values) - time_steps + 1):
        output.append(values[i : (i + time_steps)])
```

```
    return np.stack(output)


x_train = create_sequences(df_training_value.values)
print("Training input shape: ", x_train.shape)
```

```
Training input shape:  (3745, 288, 1)
```

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
#Adam
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.01), loss="mse")
model.summary()
```

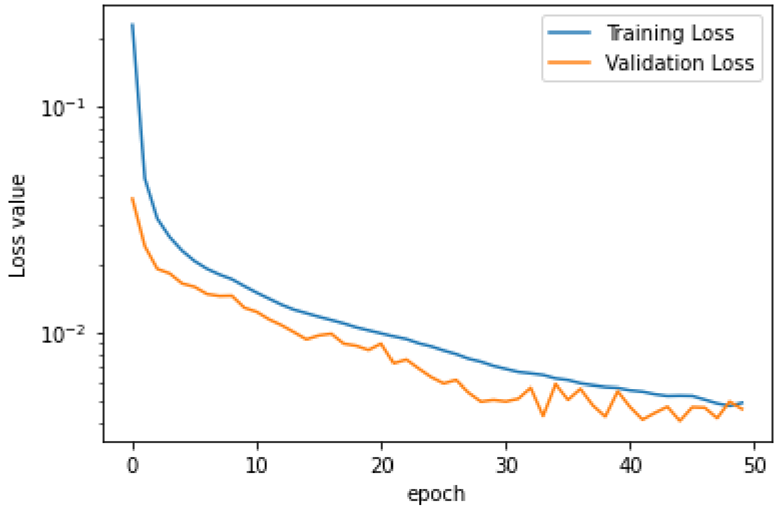
```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_28 (Conv1D)          (None, 144, 32)           256

 dropout_28 (Dropout)        (None, 144, 32)           0

 conv1d_29 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_42 (Conv1D  (None, 144, 16)          1808
 Transpose)

 dropout_29 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_43 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_44 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

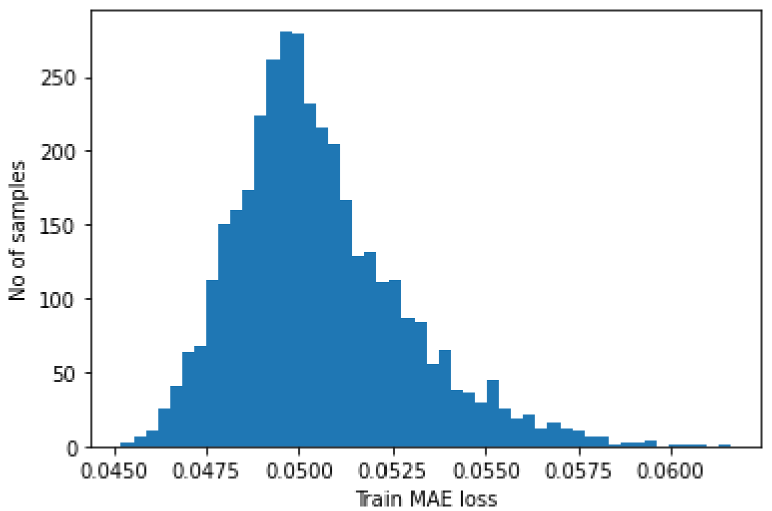
```
27/27 [==============================] - 2s 83ms/step - loss: 0.0117 - val_loss: 0.0096
Epoch 17/50
27/27 [==============================] - 3s 112ms/step - loss: 0.0113 - val_loss: 0.0098
Epoch 18/50
27/27 [==============================] - 4s 131ms/step - loss: 0.0109 - val_loss: 0.0089
Epoch 19/50
27/27 [==============================] - 3s 126ms/step - loss: 0.0105 - val_loss: 0.0087
Epoch 20/50
27/27 [==============================] - 3s 128ms/step - loss: 0.0102 - val_loss: 0.0083
Epoch 21/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0099 - val_loss: 0.0089
Epoch 22/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0096 - val_loss: 0.0073
Epoch 23/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0093 - val_loss: 0.0075
Epoch 24/50
27/27 [==============================] - 3s 126ms/step - loss: 0.0089 - val_loss: 0.0069
Epoch 25/50
27/27 [==============================] - 3s 125ms/step - loss: 0.0086 - val_loss: 0.0063
Epoch 26/50
27/27 [==============================] - 4s 142ms/step - loss: 0.0083 - val_loss: 0.0059
Epoch 27/50
27/27 [==============================] - 4s 134ms/step - loss: 0.0080 - val_loss: 0.0061
Epoch 28/50
27/27 [==============================] - 4s 132ms/step - loss: 0.0076 - val_loss: 0.0054
Epoch 29/50
27/27 [==============================] - 3s 129ms/step - loss: 0.0074 - val_loss: 0.0049
Epoch 30/50
27/27 [==============================] - 3s 123ms/step - loss: 0.0071 - val_loss: 0.0050
Epoch 31/50
27/27 [==============================] - 4s 132ms/step - loss: 0.0069 - val_loss: 0.0049
Epoch 32/50
27/27 [==============================] - 4s 131ms/step - loss: 0.0067 - val_loss: 0.0051
Epoch 33/50
27/27 [==============================] - 3s 126ms/step - loss: 0.0066 - val_loss: 0.0057
Epoch 34/50
27/27 [==============================] - 3s 124ms/step - loss: 0.0065 - val_loss: 0.0043
Epoch 35/50
27/27 [==============================] - 3s 125ms/step - loss: 0.0062 - val_loss: 0.0059
Epoch 36/50
27/27 [==============================] - 3s 127ms/step - loss: 0.0061 - val_loss: 0.0050
Epoch 37/50
27/27 [==============================] - 4s 144ms/step - loss: 0.0059 - val_loss: 0.0056
Epoch 38/50
27/27 [==============================] - 4s 130ms/step - loss: 0.0058 - val_loss: 0.0048
Epoch 39/50
27/27 [==============================] - 3s 119ms/step - loss: 0.0057 - val_loss: 0.0042
Epoch 40/50
27/27 [==============================] - 4s 135ms/step - loss: 0.0057 - val_loss: 0.0055
Epoch 41/50
27/27 [==============================] - 3s 124ms/step - loss: 0.0055 - val_loss: 0.0047
Epoch 42/50
27/27 [==============================] - 3s 118ms/step - loss: 0.0055 - val_loss: 0.0041
```

```
Epoch 43/50
27/27 [==============================] - 3s 129ms/step - loss: 0.0053 - val_loss: 0.0044
Epoch 44/50
27/27 [==============================] - 3s 99ms/step - loss: 0.0052 - val_loss: 0.0047
Epoch 45/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0052 - val_loss: 0.0041
```

```python
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```




```python
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
Reconstruction error threshold:  0.061602370242399784
```

Compare the results.

Adam - /step - loss: 0.0048 - val_loss: 0.0042

Classic SGD /step - loss: 0.0057 - val_loss: 0.0049 Адам сошёлся к лучшему результату и быстрее, а также ему требуется меньшее число samples. Класс, данный метод имеет слысл, так как даёт точность примерно на 15% лучше.

Try to perform different runs of SGD + Momentum and select the best hyperparameters. Do the same for the Adam in the similar setting. Compare the results.

```
# Normalize and save the mean and std we get,
# for normalizing test data.
training_mean = df_small_noise.mean()
training_std = df_small_noise.std()
df_training_value = (df_small_noise - training_mean) / training_std
print("Number of training samples:", len(df_training_value))
```

```
     Number of training samples: 4032
```

```
TIME_STEPS = 288

# Generated training sequences for use in the model.
def create_sequences(values, time_steps=TIME_STEPS):
    output = []
    for i in range(len(values) - time_steps + 1):
        output.append(values[i : (i + time_steps)])
    return np.stack(output)


x_train = create_sequences(df_training_value.values)
print("Training input shape: ", x_train.shape)
```

```
     Training input shape:  (3745, 288, 1)
```

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
```

```
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01), loss="mse")
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_20 (Conv1D)          (None, 144, 32)           256

 dropout_20 (Dropout)        (None, 144, 32)           0

 conv1d_21 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_30 (Conv1D  (None, 144, 16)          1808
 Transpose)

 dropout_21 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_31 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_32 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

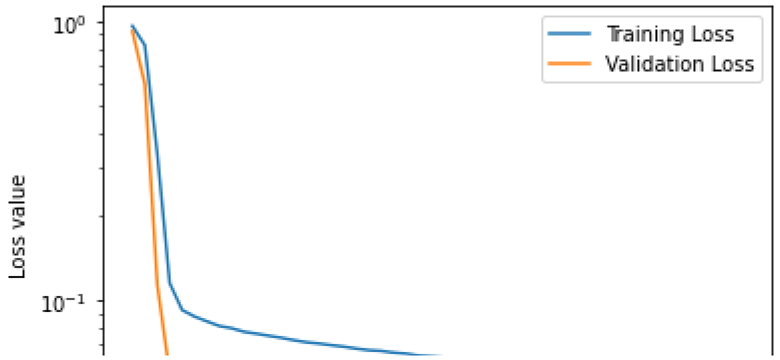
Чтобы изменить содержимое ячейки, дважды нажмите на нее (или выберите "Ввод")

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

```
27/27 [==============================] - 2s 82ms/step - loss: 0.1159 - val_loss: 0.0549
Epoch 5/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0928 - val_loss: 0.0503
Epoch 6/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0880 - val_loss: 0.0490
Epoch 7/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0845 - val_loss: 0.0481
Epoch 8/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0813 - val_loss: 0.0472
```

```
27/27 [==============================] - 2s 84ms/step - loss: 0.0815 - val_loss: 0.0472
Epoch 9/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0798 - val_loss: 0.0465
Epoch 10/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0775 - val_loss: 0.0459
Epoch 11/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0763 - val_loss: 0.0455
Epoch 12/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0750 - val_loss: 0.0450
Epoch 13/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0738 - val_loss: 0.0446
Epoch 14/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0723 - val_loss: 0.0442
Epoch 15/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0711 - val_loss: 0.0439
Epoch 16/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0704 - val_loss: 0.0436
Epoch 17/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0694 - val_loss: 0.0431
Epoch 18/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0686 - val_loss: 0.0428
Epoch 19/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0675 - val_loss: 0.0425
Epoch 20/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0666 - val_loss: 0.0424
Epoch 21/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0662 - val_loss: 0.0420
Epoch 22/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0653 - val_loss: 0.0417
Epoch 23/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0647 - val_loss: 0.0415
Epoch 24/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0636 - val_loss: 0.0414
Epoch 25/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0632 - val_loss: 0.0411
Epoch 26/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0627 - val_loss: 0.0409
Epoch 27/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0621 - val_loss: 0.0407
Epoch 28/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0613 - val_loss: 0.0405
Epoch 29/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0610 - val_loss: 0.0403
Epoch 30/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0607 - val_loss: 0.0402
Epoch 31/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0598 - val_loss: 0.0399
Epoch 32/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0595 - val_loss: 0.0398
Epoch 33/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0590 - val_loss: 0.0399
```

```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```



SGD best parameters search

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01, momentum = 0.9), loss="mse")
model.summary()
```

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_30 (Conv1D)          (None, 144, 32)           256

 dropout_30 (Dropout)        (None, 144, 32)           0

 conv1d_31 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_45 (Conv1D  (None, 144, 16)          1808
 Transpose)

 dropout_31 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_46 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_47 (Conv1D  (None, 288, 1)           225
```

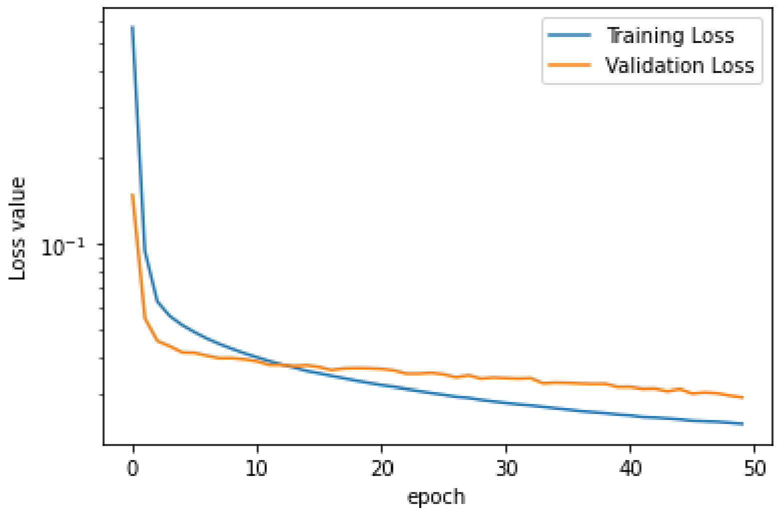
```
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

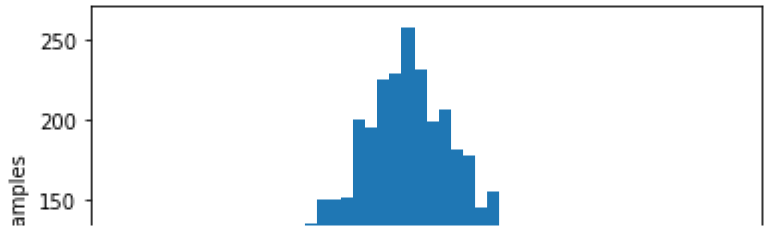
```
27/27 [==============================] - 2s 84ms/step - loss: 0.0353 - val_loss: 0.0371
Epoch 17/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0346 - val_loss: 0.0362
Epoch 18/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0339 - val_loss: 0.0367
Epoch 19/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0332 - val_loss: 0.0368
Epoch 20/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0327 - val_loss: 0.0367
Epoch 21/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0321 - val_loss: 0.0366
Epoch 22/50

27/27 [==============================] - 2s 84ms/step - loss: 0.0317 - val_loss: 0.0361
Epoch 23/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0311 - val_loss: 0.0352
Epoch 24/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0307 - val_loss: 0.0352
Epoch 25/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0302 - val_loss: 0.0354
Epoch 26/50
27/27 [==============================] - 3s 107ms/step - loss: 0.0298 - val_loss: 0.0350
Epoch 27/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0293 - val_loss: 0.0342
Epoch 28/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0290 - val_loss: 0.0347
Epoch 29/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0286 - val_loss: 0.0338
Epoch 30/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0282 - val_loss: 0.0341
Epoch 31/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0279 - val_loss: 0.0339
Epoch 32/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0276 - val_loss: 0.0338
Epoch 33/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0273 - val_loss: 0.0339
Epoch 34/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0270 - val_loss: 0.0326
Epoch 35/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0267 - val_loss: 0.0327
Epoch 36/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0264 - val_loss: 0.0327
Epoch 37/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0261 - val_loss: 0.0325
Epoch 38/50
```

```
27/27 [==============================] - 2s 84ms/step - loss: 0.0259 - val_loss: 0.0324
Epoch 39/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0257 - val_loss: 0.0324
Epoch 40/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0254 - val_loss: 0.0316
Epoch 41/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0252 - val_loss: 0.0317
Epoch 42/50
27/27 [==============================] - 4s 146ms/step - loss: 0.0249 - val_loss: 0.0312
Epoch 43/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0248 - val_loss: 0.0312
Epoch 44/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0246 - val_loss: 0.0306
Epoch 45/50
```

```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```




```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01, momentum = 0.8), loss="mse")
model.summary()
```

```
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_32 (Conv1D)          (None, 144, 32)           256

 dropout_32 (Dropout)        (None, 144, 32)           0

 conv1d_33 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_48 (Conv1D  (None, 144, 16)          1808
 Transpose)

 dropout_33 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_49 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_50 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

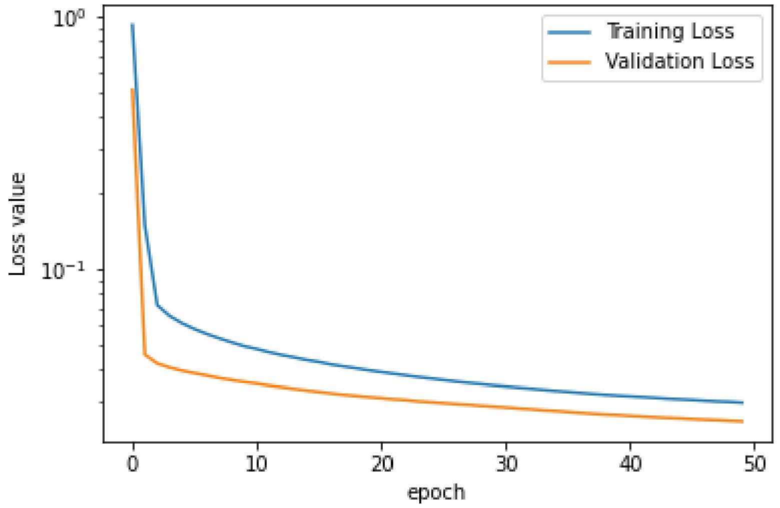
```
history = model.fit(
    x_train,
```

```
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

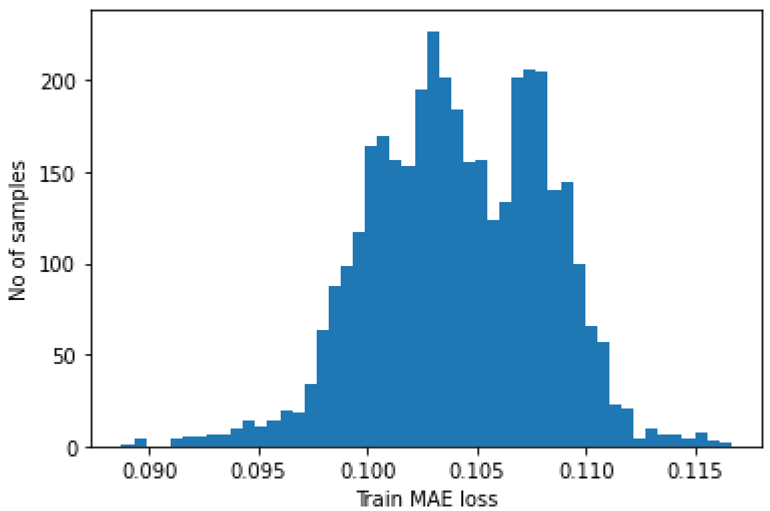
```
Epoch 1/50
27/27 [==============================] - 3s 89ms/step - loss: 0.9270 - val_loss: 0.5128
Epoch 2/50
27/27 [==============================] - 2s 82ms/step - loss: 0.1497 - val_loss: 0.0459
Epoch 3/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0721 - val_loss: 0.0424
Epoch 4/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0653 - val_loss: 0.0409
Epoch 5/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0611 - val_loss: 0.0397
Epoch 6/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0580 - val_loss: 0.0388
Epoch 7/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0554 - val_loss: 0.0380
Epoch 8/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0532 - val_loss: 0.0371
Epoch 9/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0513 - val_loss: 0.0365
Epoch 10/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0496 - val_loss: 0.0358
Epoch 11/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0483 - val_loss: 0.0353
Epoch 12/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0469 - val_loss: 0.0347
Epoch 13/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0458 - val_loss: 0.0342
Epoch 14/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0448 - val_loss: 0.0336
Epoch 15/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0438 - val_loss: 0.0332
Epoch 16/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0430 - val_loss: 0.0327
Epoch 17/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0420 - val_loss: 0.0323
Epoch 18/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0413 - val_loss: 0.0319
Epoch 19/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0405 - val_loss: 0.0316
Epoch 20/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0398 - val_loss: 0.0312
Epoch 21/50
27/27 [==============================] - 3s 107ms/step - loss: 0.0392 - val_loss: 0.0310
Epoch 22/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0386 - val_loss: 0.0306
Epoch 23/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0380 - val_loss: 0.0304
Epoch 24/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0374 - val_loss: 0.0300
Epoch 25/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0370 - val_loss: 0.0298
Epoch 26/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0365 - val_loss: 0.0296
Epoch 27/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0360 - val_loss: 0.0293
```

```
Epoch 28/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0355 - val_loss: 0.0291
Epoch 29/50
```

```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```




```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
Reconstruction error threshold:  0.11666564462503078
```

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
```

```
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01, momentum = 0.5), loss="mse")
model.summary()
```

```
Model: "sequential_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_34 (Conv1D)          (None, 144, 32)           256

 dropout_34 (Dropout)        (None, 144, 32)           0

 conv1d_35 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_51 (Conv1D  (None, 144, 16)          1808
 Transpose)

 dropout_35 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_52 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_53 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

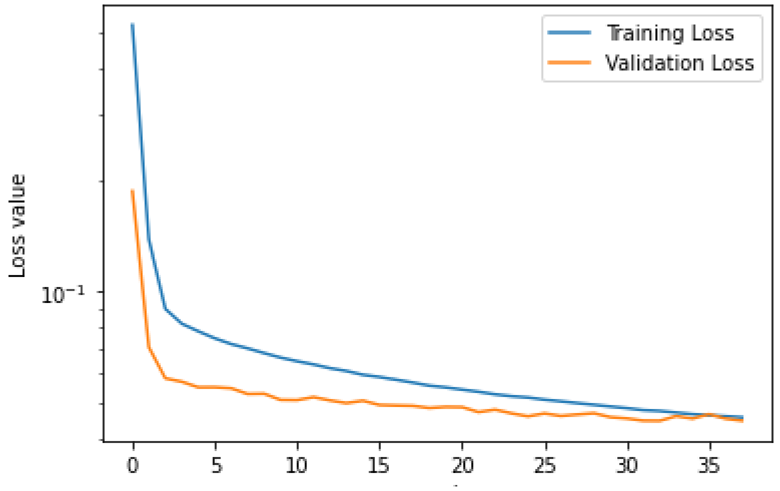
```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

```
27/27 [==============================] - 2s 81ms/step - loss: 0.0720 - val_loss: 0.0547
Epoch 8/50
```

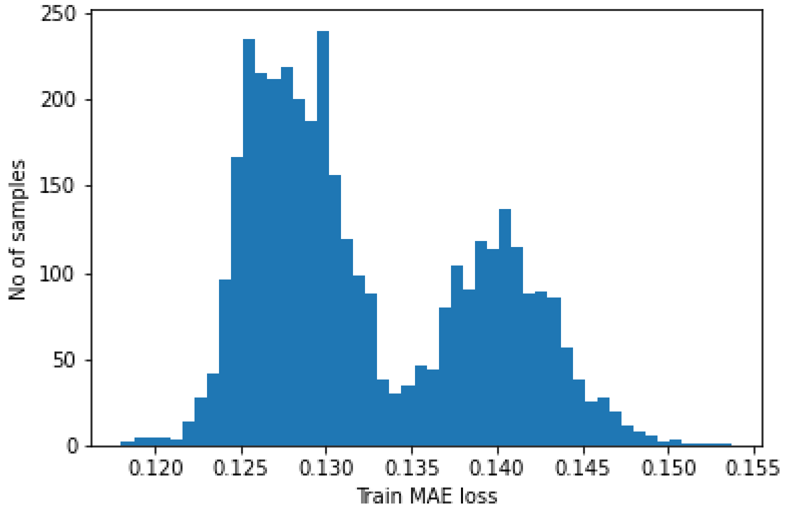
```
27/27 [==============================] - 2s 81ms/step - loss: 0.0701 - val_loss: 0.0528
Epoch 9/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0681 - val_loss: 0.0528
Epoch 10/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0662 - val_loss: 0.0508
Epoch 11/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0647 - val_loss: 0.0508
Epoch 12/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0634 - val_loss: 0.0517
Epoch 13/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0620 - val_loss: 0.0507
Epoch 14/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0609 - val_loss: 0.0498
Epoch 15/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0595 - val_loss: 0.0506
Epoch 16/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0586 - val_loss: 0.0492
Epoch 17/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0576 - val_loss: 0.0491
Epoch 18/50
27/27 [==============================] - 3s 107ms/step - loss: 0.0566 - val_loss: 0.0490
Epoch 19/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0556 - val_loss: 0.0483
Epoch 20/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0550 - val_loss: 0.0487
Epoch 21/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0542 - val_loss: 0.0486
Epoch 22/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0535 - val_loss: 0.0471
Epoch 23/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0526 - val_loss: 0.0478
Epoch 24/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0520 - val_loss: 0.0467
Epoch 25/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0516 - val_loss: 0.0459
Epoch 26/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0509 - val_loss: 0.0467
Epoch 27/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0504 - val_loss: 0.0460
Epoch 28/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0498 - val_loss: 0.0464
Epoch 29/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0493 - val_loss: 0.0468
Epoch 30/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0488 - val_loss: 0.0456
Epoch 31/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0483 - val_loss: 0.0453
Epoch 32/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0477 - val_loss: 0.0447
Epoch 33/50

27/27 [==============================] - 2s 81ms/step - loss: 0.0474 - val_loss: 0.0446
Epoch 34/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0470 - val_loss: 0.0459
Epoch 35/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0465 - val_loss: 0.0452
Epoch 36/50
```

```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
```

```
plt.legend()
plt.show()
```




```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
Reconstruction error threshold:  0.1537037494491309
```

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
```

```
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01, momentum = 0.3), loss="mse")
model.summary()
```

```
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_36 (Conv1D)          (None, 144, 32)           256

 dropout_36 (Dropout)        (None, 144, 32)           0

 conv1d_37 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_54 (Conv1D  (None, 144, 16)          1808
 Transpose)

 dropout_37 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_55 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_56 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

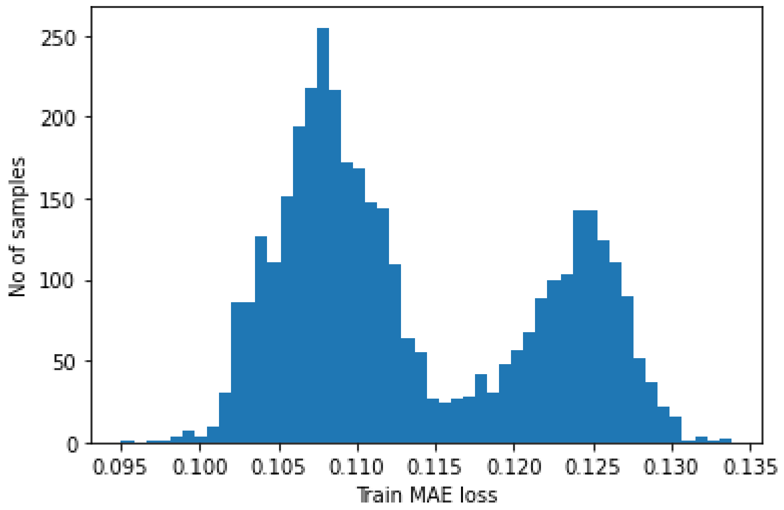
```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

```
27/27 [==============================] - 2s 83ms/step - loss: 0.0667 - val_loss: 0.0467
Epoch 17/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0654 - val_loss: 0.0461
Epoch 18/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0645 - val_loss: 0.0454
Epoch 19/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0634 - val_loss: 0.0450
Epoch 20/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0625 - val_loss: 0.0444
Epoch 21/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0616 - val_loss: 0.0439
Epoch 22/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0606 - val_loss: 0.0434
```

```
Epoch 23/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0599 - val_loss: 0.0429
Epoch 24/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0592 - val_loss: 0.0424
Epoch 25/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0583 - val_loss: 0.0413
Epoch 26/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0575 - val_loss: 0.0413
Epoch 27/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0569 - val_loss: 0.0406
Epoch 28/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0563 - val_loss: 0.0402
Epoch 29/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0556 - val_loss: 0.0399
Epoch 30/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0549 - val_loss: 0.0389
Epoch 31/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0544 - val_loss: 0.0394
Epoch 32/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0539 - val_loss: 0.0388
Epoch 33/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0532 - val_loss: 0.0391
Epoch 34/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0527 - val_loss: 0.0388
Epoch 35/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0520 - val_loss: 0.0376
Epoch 36/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0515 - val_loss: 0.0378
Epoch 37/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0510 - val_loss: 0.0375
Epoch 38/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0508 - val_loss: 0.0373

Epoch 39/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0502 - val_loss: 0.0372
Epoch 40/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0498 - val_loss: 0.0368
Epoch 41/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0495 - val_loss: 0.0368
Epoch 42/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0490 - val_loss: 0.0358
Epoch 43/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0486 - val_loss: 0.0365
Epoch 44/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0481 - val_loss: 0.0355
Epoch 45/50
```

```python
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```


```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
Reconstruction error threshold:  0.1337828755817874
```

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01, momentum = 0.81), loss="mse")
model.summary()
```

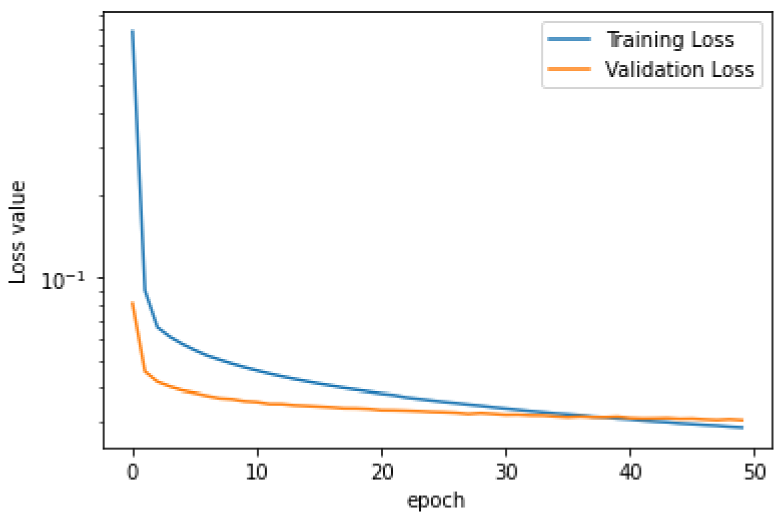
```
Model: "sequential_21"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_42 (Conv1D)          (None, 144, 32)           256

 dropout_42 (Dropout)        (None, 144, 32)           0

 conv1d_43 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_63 (Conv1D  (None, 144, 16)          1808
 Transpose)

 dropout_43 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_64 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_65 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

```
27/27 [==============================] - 3s 113ms/step - loss: 0.0412 - val_loss: 0.0341
Epoch 17/50
27/27 [==============================] - 3s 106ms/step - loss: 0.0405 - val_loss: 0.0339
Epoch 18/50
27/27 [==============================] - 3s 117ms/step - loss: 0.0398 - val_loss: 0.0337
Epoch 19/50
27/27 [==============================] - 3s 117ms/step - loss: 0.0392 - val_loss: 0.0336
Epoch 20/50
27/27 [==============================] - 3s 122ms/step - loss: 0.0386 - val_loss: 0.0335
Epoch 21/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0380 - val_loss: 0.0331
Epoch 22/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0375 - val_loss: 0.0331
Epoch 23/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0368 - val_loss: 0.0330
Epoch 24/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0364 - val_loss: 0.0329
Epoch 25/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0360 - val_loss: 0.0327
Epoch 26/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0355 - val_loss: 0.0326
Epoch 27/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0352 - val_loss: 0.0324
Epoch 28/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0347 - val_loss: 0.0321
```

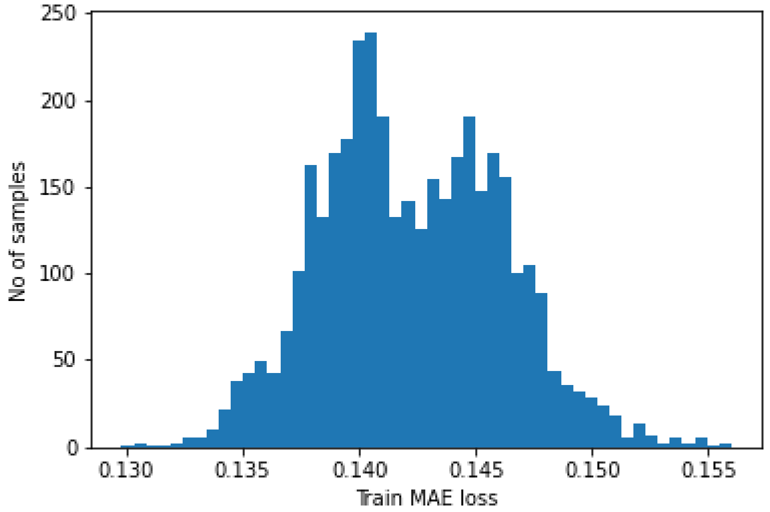
```
Epoch 29/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0344 - val_loss: 0.0324
Epoch 30/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0340 - val_loss: 0.0321
Epoch 31/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0336 - val_loss: 0.0319
Epoch 32/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0333 - val_loss: 0.0319
Epoch 33/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0329 - val_loss: 0.0318
Epoch 34/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0326 - val_loss: 0.0317
Epoch 35/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0323 - val_loss: 0.0315
Epoch 36/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0320 - val_loss: 0.0313
Epoch 37/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0318 - val_loss: 0.0314
Epoch 38/50
27/27 [==============================] - 2s 79ms/step - loss: 0.0314 - val_loss: 0.0313
Epoch 39/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0312 - val_loss: 0.0313
Epoch 40/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0309 - val_loss: 0.0314
Epoch 41/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0307 - val_loss: 0.0310
Epoch 42/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0304 - val_loss: 0.0309
Epoch 43/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0301 - val_loss: 0.0310
Epoch 44/50
27/27 [==============================] - 2s 80ms/step - loss: 0.0300 - val_loss: 0.0310

Epoch 45/50
```

```python
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```




```python
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)
```

```
plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
Reconstruction error threshold:  0.15598558112817093
```

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=keras.optimizers.SGD(learning_rate=0.01, momentum = 0.79), loss="mse")
model.summary()
```

```
Model: "sequential_22"

_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv1d_44 (Conv1D)          (None, 144, 32)           256       

 dropout_44 (Dropout)        (None, 144, 32)           0         

 conv1d_45 (Conv1D)          (None, 72, 16)            3600      
```

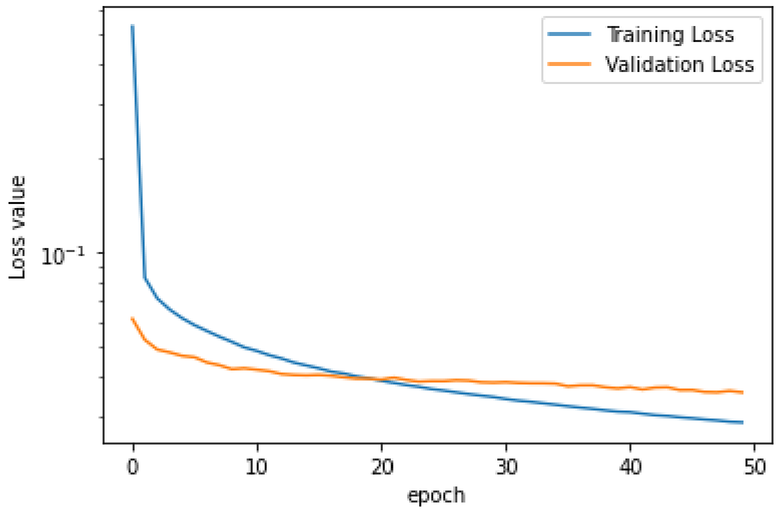
```
 conv1d_transpose_66 (Conv1D  (None, 144, 16)          1808
 Transpose)

 dropout_45 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_67 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_68 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

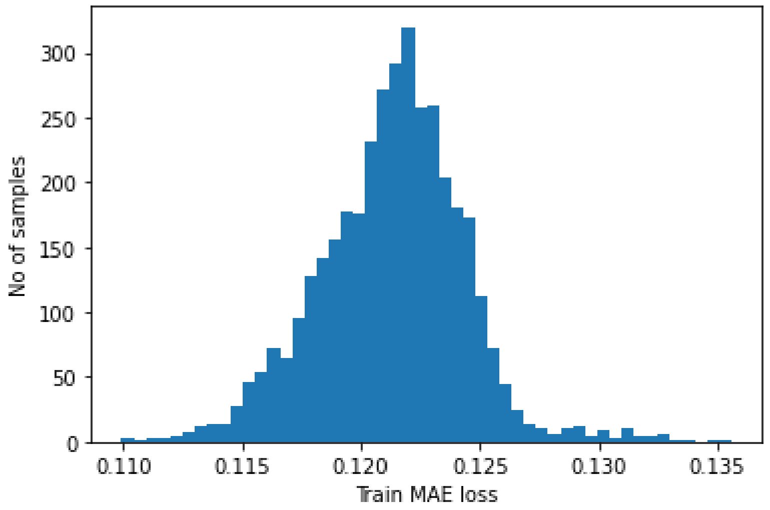
```
Epoch 2/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0831 - val_loss: 0.0526
Epoch 3/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0714 - val_loss: 0.0489
Epoch 4/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0657 - val_loss: 0.0479
Epoch 5/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0617 - val_loss: 0.0467
Epoch 6/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0586 - val_loss: 0.0462
Epoch 7/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0561 - val_loss: 0.0445
Epoch 8/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0538 - val_loss: 0.0436
Epoch 9/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0518 - val_loss: 0.0423
Epoch 10/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0497 - val_loss: 0.0426
Epoch 11/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0484 - val_loss: 0.0422
Epoch 12/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0470 - val_loss: 0.0417
Epoch 13/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0458 - val_loss: 0.0408
Epoch 14/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0444 - val_loss: 0.0406
Epoch 15/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0435 - val_loss: 0.0405
Epoch 16/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0426 - val_loss: 0.0406
Epoch 17/50

27/27 [==============================] - 2s 82ms/step - loss: 0.0416 - val_loss: 0.0403
Epoch 18/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0410 - val_loss: 0.0399
```

```
Epoch 19/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0402 - val_loss: 0.0395
Epoch 20/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0396 - val_loss: 0.0395
Epoch 21/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0389 - val_loss: 0.0392
Epoch 22/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0383 - val_loss: 0.0398
Epoch 23/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0376 - val_loss: 0.0391
Epoch 24/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0372 - val_loss: 0.0386
Epoch 25/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0366 - val_loss: 0.0388
Epoch 26/50
27/27 [==============================] - 2s 82ms/step - loss: 0.0362 - val_loss: 0.0388
Epoch 27/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0357 - val_loss: 0.0390
Epoch 28/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0352 - val_loss: 0.0389
Epoch 29/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0348 - val_loss: 0.0384
Epoch 30/50
27/27 [==============================] - 2s 81ms/step - loss: 0.0345 - val_loss: 0.0383
```

```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```




```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
Reconstruction error threshold:  0.1355820500003613
```

Оптимизируем Adam

```
"""
tf.keras.optimizers.Adam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=False,
    name="Adam",
    **kwargs
)
"""


model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
#Adam
model.compile(optimizer=keras.optimizers.Adam(
    learning_rate=0.01,
    beta_1=0.8,
```

```
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=False,
    name="Adam"
), loss="mse")
model.summary()
```

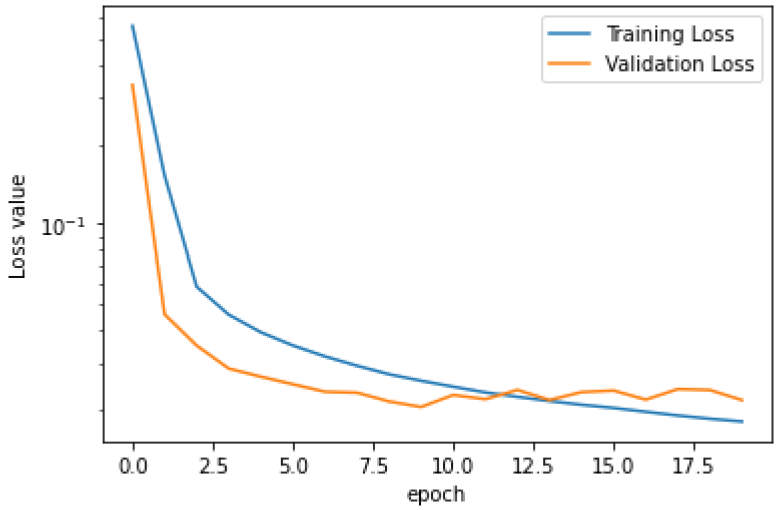
```
Model: "sequential_27"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv1d_54 (Conv1D)          (None, 144, 32)           256       
                                                                 
 dropout_54 (Dropout)        (None, 144, 32)           0         
                                                                 
 conv1d_55 (Conv1D)          (None, 72, 16)            3600      
                                                                 
 conv1d_transpose_81 (Conv1D  (None, 144, 16)          1808      
 Transpose)                                                      
                                                                 
 dropout_55 (Dropout)        (None, 144, 16)           0         
                                                                 
 conv1d_transpose_82 (Conv1D  (None, 288, 32)          3616      
 Transpose)                                                      
                                                                 
 conv1d_transpose_83 (Conv1D  (None, 288, 1)           225       
 Transpose)                                                      
                                                                 
=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience= 10, mode="min")
    ],
)
```

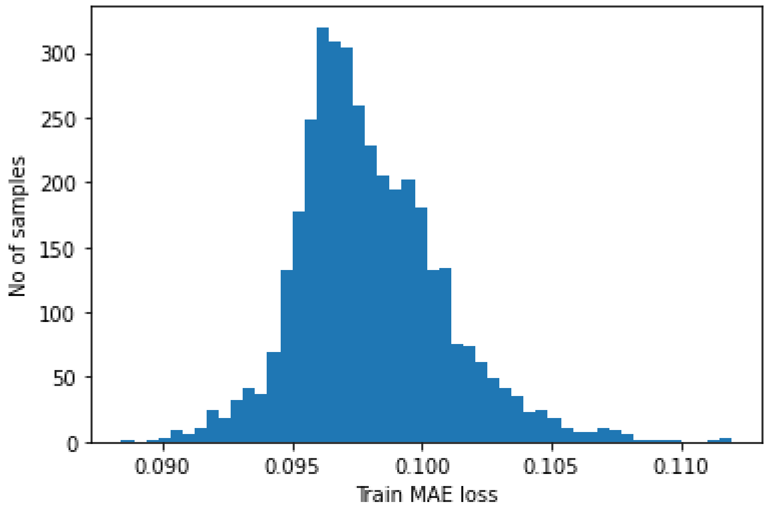
```
Epoch 1/50
27/27 [==============================] - 7s 190ms/step - loss: 0.5615 - val_loss: 0.3357
Epoch 2/50
27/27 [==============================] - 4s 158ms/step - loss: 0.1527 - val_loss: 0.0459
Epoch 3/50
27/27 [==============================] - 4s 147ms/step - loss: 0.0584 - val_loss: 0.0350
Epoch 4/50
27/27 [==============================] - 4s 142ms/step - loss: 0.0458 - val_loss: 0.0286
Epoch 5/50
27/27 [==============================] - 4s 136ms/step - loss: 0.0393 - val_loss: 0.0267
Epoch 6/50
27/27 [==============================] - 3s 115ms/step - loss: 0.0350 - val_loss: 0.0250
Epoch 7/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0318 - val_loss: 0.0234
Epoch 8/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0293 - val_loss: 0.0232
Epoch 9/50
```

```
27/27 [==============================] - 2s 84ms/step - loss: 0.0272 - val_loss: 0.0215
Epoch 10/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0257 - val_loss: 0.0205
Epoch 11/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0244 - val_loss: 0.0227
Epoch 12/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0232 - val_loss: 0.0219
Epoch 13/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0224 - val_loss: 0.0237
Epoch 14/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0216 - val_loss: 0.0217
Epoch 15/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0209 - val_loss: 0.0233
Epoch 16/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0203 - val_loss: 0.0236
Epoch 17/50
27/27 [==============================] - 2s 86ms/step - loss: 0.0196 - val_loss: 0.0218
Epoch 18/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0190 - val_loss: 0.0239
Epoch 19/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0185 - val_loss: 0.0237
Epoch 20/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0180 - val_loss: 0.0217
```

```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```




```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```




```
Reconstruction error threshold:  0.11194248254913577
```

```
model = keras.Sequential(
    [
        layers.Input(shape=(x_train.shape[1], x_train.shape[2])),
        layers.Conv1D(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=7, padding="same", strides=2, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=7, padding="same"),
    ]
)
#optimizer = keras.optimizers.SGD(learning_rate=0.01)
#Adam
model.compile(optimizer=keras.optimizers.Adam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=True,
    name="Adam"
), loss="mse")
model.summary()
```

```
Model: "sequential_29"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_58 (Conv1D)          (None, 144, 32)           256

 dropout_58 (Dropout)        (None, 144, 32)           0

 conv1d_59 (Conv1D)          (None, 72, 16)            3600

 conv1d_transpose_87 (Conv1D  (None, 144, 16)          1808
 Transpose)
```

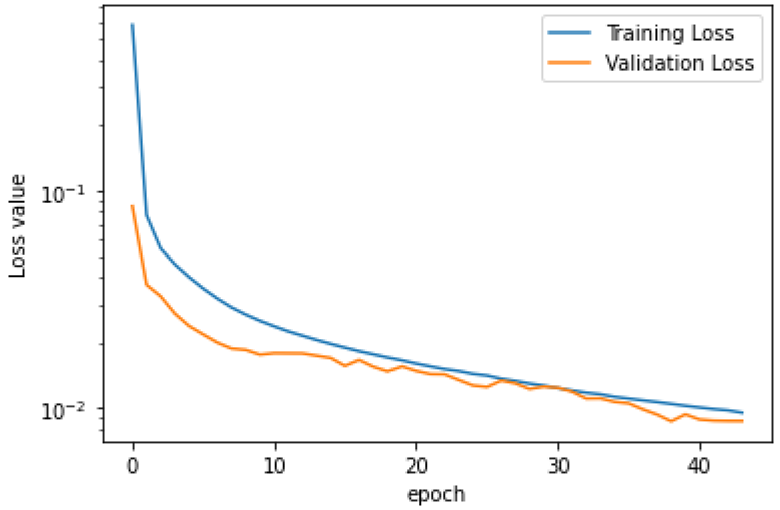
```
 dropout_59 (Dropout)        (None, 144, 16)           0

 conv1d_transpose_88 (Conv1D  (None, 288, 32)          3616
 Transpose)

 conv1d_transpose_89 (Conv1D  (None, 288, 1)           225
 Transpose)

=================================================================
Total params: 9,505
Trainable params: 9,505
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(
    x_train,
    x_train,
    epochs=50,
    batch_size=128,
    validation_split=0.1,
    callbacks=[
        keras.callbacks.EarlyStopping(monitor="val_loss", patience=5, mode="min")
    ],
)
```

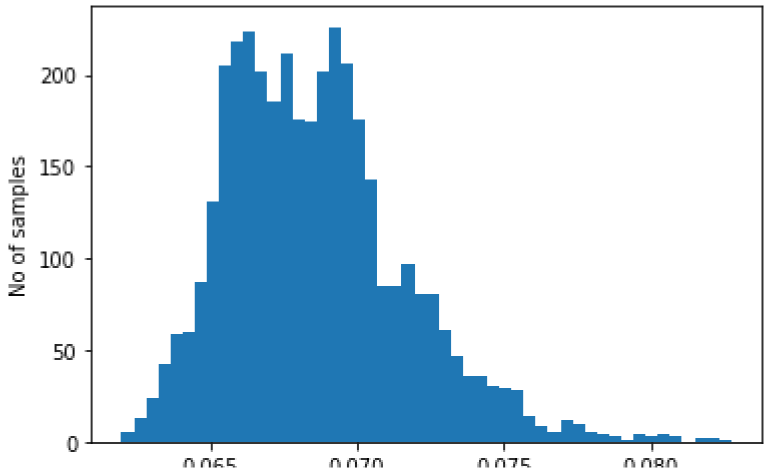
```
Epoch 12/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0225 - val_loss: 0.0179
Epoch 13/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0215 - val_loss: 0.0178
Epoch 14/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0206 - val_loss: 0.0174
Epoch 15/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0197 - val_loss: 0.0170
Epoch 16/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0190 - val_loss: 0.0156
Epoch 17/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0183 - val_loss: 0.0166
Epoch 18/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0177 - val_loss: 0.0155

Epoch 19/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0171 - val_loss: 0.0147
Epoch 20/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0166 - val_loss: 0.0155
Epoch 21/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0160 - val_loss: 0.0148
Epoch 22/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0155 - val_loss: 0.0143
Epoch 23/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0151 - val_loss: 0.0143
Epoch 24/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0148 - val_loss: 0.0135
Epoch 25/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0144 - val_loss: 0.0127
Epoch 26/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0141 - val_loss: 0.0125
Epoch 27/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0136 - val_loss: 0.0134
Epoch 28/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0133 - val_loss: 0.0130
Epoch 29/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0130 - val_loss: 0.0122
Epoch 30/50
```

```
27/27 [==============================] - 2s 83ms/step - loss: 0.0127 - val_loss: 0.0125
Epoch 31/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0123 - val_loss: 0.0124
Epoch 32/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0120 - val_loss: 0.0119
Epoch 33/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0118 - val_loss: 0.0111
Epoch 34/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0116 - val_loss: 0.0111
Epoch 35/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0112 - val_loss: 0.0107
Epoch 36/50
27/27 [==============================] - 2s 84ms/step - loss: 0.0110 - val_loss: 0.0105
Epoch 37/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0108 - val_loss: 0.0099
Epoch 38/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0106 - val_loss: 0.0093
Epoch 39/50
27/27 [==============================] - 2s 83ms/step - loss: 0.0104 - val_loss: 0.0087
Epoch 40/50
27/27 [==============================] - 2s 85ms/step - loss: 0.0102 - val_loss: 0.0093
```

```
plt.semilogy(history.history["loss"], label="Training Loss")
plt.semilogy(history.history["val_loss"], label="Validation Loss")
plt.ylabel('Loss value')
plt.xlabel('epoch')
plt.legend()
plt.show()
```




```
# Get train MAE loss.
x_train_pred = model.predict(x_train)
train_mae_loss = np.mean(np.abs(x_train_pred - x_train), axis=1)

plt.hist(train_mae_loss, bins=50)
plt.xlabel("Train MAE loss")
plt.ylabel("No of samples")
plt.show()

# Get reconstruction loss threshold.
threshold = np.max(train_mae_loss)
print("Reconstruction error threshold: ", threshold)
```



Try to perform different runs of SGD + Momentum and select the best hyperparameters. Do the same for the Adam in the similar setting. Compare the results.

Optional: Log results with wandb
Для SGD с моментумом имеем лучший результат val loss для momentum = 0.8 loss: 0.0297 - val_loss: 0.0251

Для 0.79 loss: 0.0286 - val_loss: 0.0357

и 0.81 loss: 0.0287 - val_loss: 0.0305 хотя их train loss ниже.

Для остальных значений оно выходит за 0.03

Для Adam базовое значение равно: loss: 0.0049 - val_loss: 0.0046

С beta1 = 0.8: loss: 0.0073 - val_loss: 0.0082

Однако если включить amsgrad, то step - loss: 0.0095 - val_loss: 0.0087

Вывод: адам, конечно работает быстрее momentum и точнее. В 5 раз. И уж точно быстрее базового sgd с (loss: 0.0525 - val_loss: 0.0369), давая точность выше в 7 раз.
Другими словами, эффективность использования более сложных и продвинутых алгоритмов очевидна, так как в реальности разброс в 1 метр позволяет попасть в ростовую мишень, а разброс в 5 или 7 - нет.
Времена работы в целом одинаковые и нельзя сказать, что кто-то работает даже на 5-10% процентов быстрее другого, но это специфика задачи.

Log results with wandb - сделано.

Try to perform different runs of SGD + Momentum and select the best hyperparameters. Do the same for the Adam in the similar setting. Compare the results.

Optional: Log results with wandb Для SGD с моментумом имеем лучший результат val loss для momentum = 0.8 loss: 0.0297 - val_loss: 0.0251

Для 0.79 loss: 0.0286 - val_loss: 0.0357

и 0.81 loss: 0.0287 - val_loss: 0.0305 хотя их train loss ниже.

Для остальных значений оно выходит за 0.03

Для Adam базовое значение равно: loss: 0.0049 - val_loss: 0.0046

С beta1 = 0.8: loss: 0.0073 - val_loss: 0.0082

Однако если включить amsgrad, то step - loss: 0.0095 - val_loss: 0.0087

Вывод: адам, конечно работает быстрее momentum и точнее. В 5 раз. И уж точно быстрее базового sgd с (loss: 0.0525 - val_loss: 0.0369), давая точность выше в 7 раз. Другими словами, эффективность использования более сложных и продвинутых алгоритмов очевидна, так как в реальности разброс в 1 метр позволяет попасть в ростовую мишень, а разброс в 5 или 7 - нет. Времена работы в целом одинаковые и нельзя сказать, что кто-то работает даже на 5-10% процентов быстрее другого, но это специфика задачи.

Log results with wandb - сделано.

+ Код + Текст

Link to the keras + wandb.

```
%%capture
!pip install wandb
```

```
import wandb
from wandb.keras import WandbCallback

!wandb login
```

wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit:
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc

```
# Initialize wandb with your project name
run = wandb.init(project='my-keras-integration',
                 config={  # and include hyperparameters and metadata
                     "learning_rate": 0.01,
                     "epochs": 50,
                     "batch_size": 128,
                     "loss_function": "mse",
                     "architecture": "Autoencoder",
                     "dataset": "Daily anomaly"
                 })
config = wandb.config  # We'll use this to configure our experiment
```

wandb: Currently logged in as: hcl-271 (g_party). Use `wandb login --relogin` to force relogin
Tracking run with wandb version 0.12.16
Run data is saved locally in /content/wandb/run-20220508_170759-2hqjl2s8
Syncing run **playful-sky-1** to Weights & Biases (docs)