Свернули-развернули: TorchCNNBuilder для прикладных задач

Борисова Юлия
м.н.с. Лаборатория композитного искусственного интеллекта
Университет ИТМО

jul.borisova@itmo.ru

Open Source Meetup 2024,
8 октября 2024, Санкт-Петербург

Библиотека (1/2)

TorchCNNBuilder – помощник в сборке сверточных нейронных сетей

README BSD-3-Clause license

TorchCNNBuilder

pytorch | python 3.8 | 3.9 | 3.10 | 3.11 | 3.12 | pypi package 0.0.19

TorchCNNBuilder is an open-source framework for the automatic creation of CNN architectures. This framework should first of all help researchers in the applicability of CNN models for a huge range of tasks, taking over most of the writing of the architecture code. This framework is distributed under the 3-Clause BSD license. All the functionality is written only using `pytorch` *(no third-party dependencies)*

Installation

The simplest way to install framework is using `pip` :

```
pip install torchcnnbuilder
```

About

Framework for the automatic creation of CNN architectures

open-source time-series cv torch cnn-architecture

Readme
BSD-3-Clause license
Activity
26 stars
5 watching
1 fork

Releases 4

v0.1.3 Latest
on Aug 19

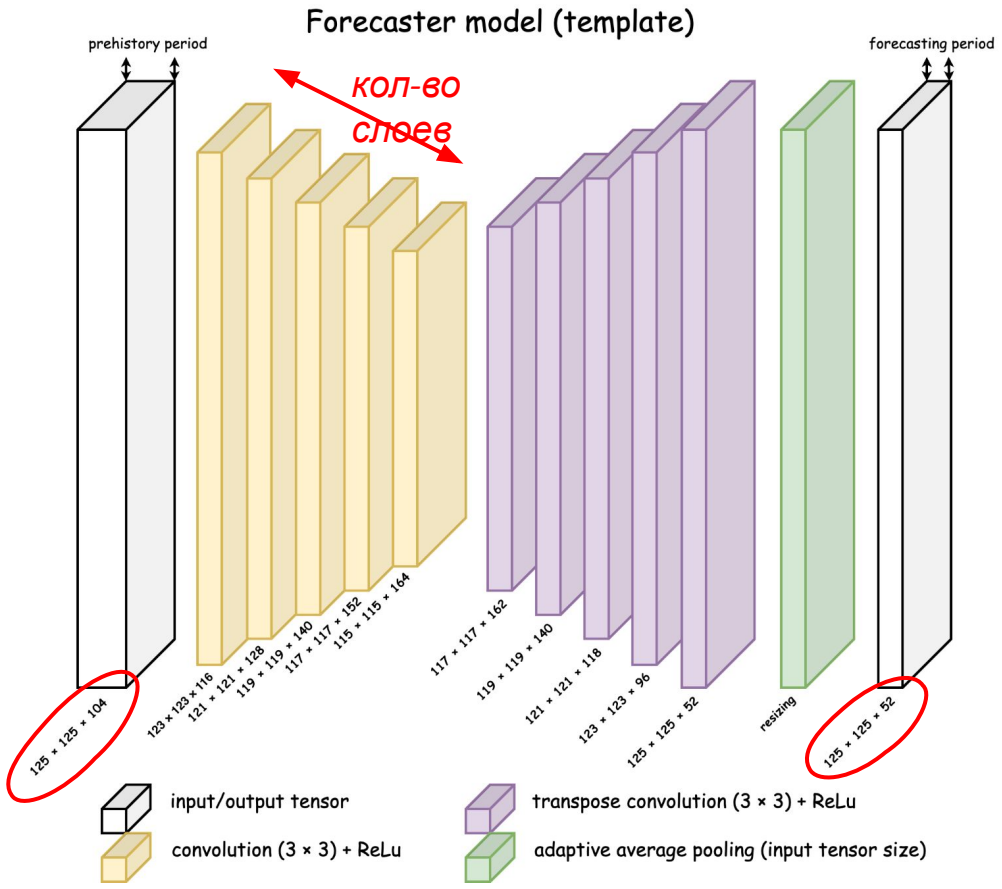
https://github.com/ChrisLisbon/TorchCNNBuilder

30 лет ФОНД СОДЕЙСТВИЯ ИННОВАЦИЯМ

Библиотека для автоматизации генерации сверточных нейронных сетей для задач предсказательного моделирования природных сред

КОД-ИИ VI/ Перспективные методы искусственного интеллекта

Библиотека (2/2) – что делает инструмент?



Листинг кода, осуществляющего сборку модели и вывод ее структуры

```
model = ForecasterBase(input_size=(125, 125),
                       n_layers=5,
                       in_time_points=104,
                       out_time_points=52)
print(model)
```

Структура собранной модели в нотации PyTorch

```
ForecasterBase(
  (convolve): Sequential(
    (conv1): Sequential(
      (0): Conv2d(104, 116, kernel_size=(3, 3), stride=(1, 1))
      (1): ReLU(inplace=True)
    )
    (conv2): Sequential(
      (0): Conv2d(116, 128, kernel_size=(3, 3), stride=(1, 1))
      (1): ReLU(inplace=True)
    )
    (conv3): Sequential(
      (0): Conv2d(128, 140, kernel_size=(3, 3), stride=(1, 1))
      (1): ReLU(inplace=True)
    )
    (conv4): Sequential(
      (0): Conv2d(140, 152, kernel_size=(3, 3), stride=(1, 1))
      (1): ReLU(inplace=True)
    )
    (conv5): Sequential(
      (0): Conv2d(152, 164, kernel_size=(3, 3), stride=(1, 1))
      (1): ReLU(inplace=True)
    )
  )
  (transpose): Sequential(
    (deconv1): Sequential(
      (0): ConvTranspose2d(164, 162, kernel_size=(3, 3), stride=(1, 1))
      (1): ReLU(inplace=True)
    )
    (deconv2): Sequential(
      (0): ConvTranspose2d(162, 140, kernel_size=(3, 3), stride=(1, 1))
      (1): ReLU(inplace=True)
    )
    (deconv3): Sequential(
      (0): ConvTranspose2d(140, 118, kernel_size=(3, 3), stride=(1, 1))
      (1): ReLU(inplace=True)
    )
    (deconv4): Sequential(
      (0): ConvTranspose2d(118, 96, kernel_size=(3, 3), stride=(1, 1))
      (1): ReLU(inplace=True)
    )
    (deconv5): Sequential(
      (0): ConvTranspose2d(96, 52, kernel_size=(3, 3), stride=(1, 1))
    )
    (resize): AdaptiveAvgPool2d(output_size=(125, 125))
  )
)
```

Последовательность слоев прямых сверток, блок кодировщик

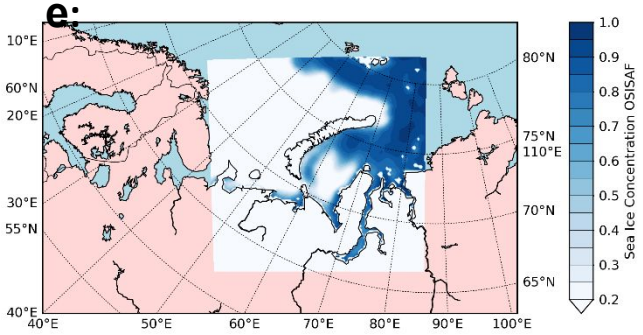
Последовательность слоев обратных сверток, блок декодировщик

Слой пуллинга для сохранения размерностей

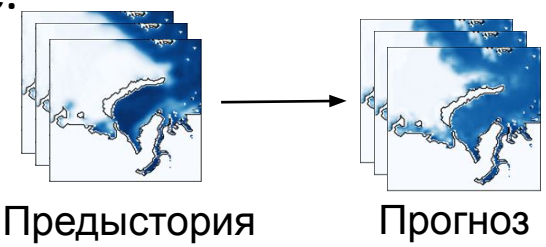
Стартовая идея

Пространственно-временное моделирование (прогноз природных полей)

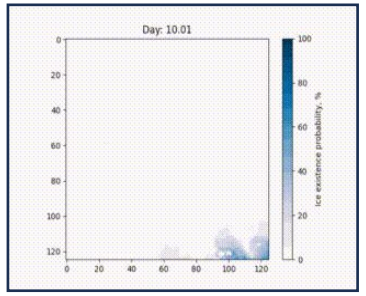
Данные:



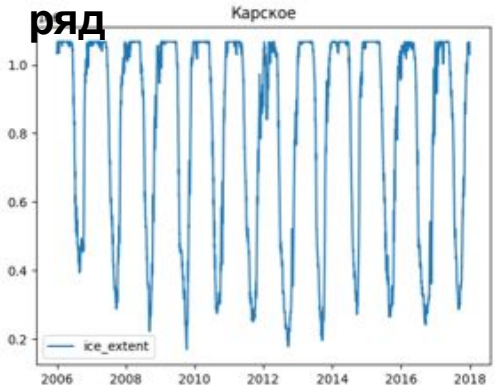
Ожидание:



Динамика:



В точке – временной ряд



На сетке в пространстве – пространственно-временной ряд

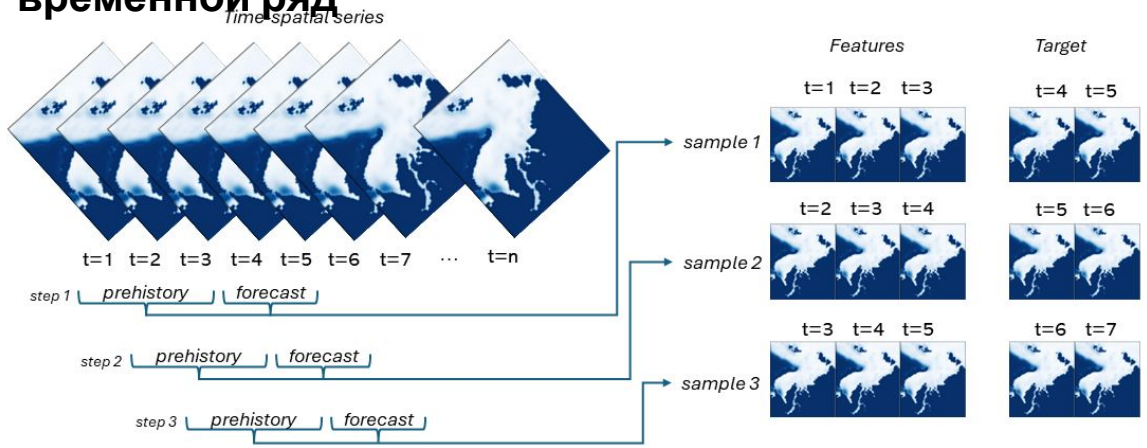


Схема предподготовки данных, реализована в TorchCNNBuilder

Подготовка данных – аналог lagged-преобразования для временных рядов. Реализовано скользящим окном по всему ряду.

```
dataset = multi_output_tensor(
    data=data,
    forecast_len=30,
    pre_history_len=60)
```

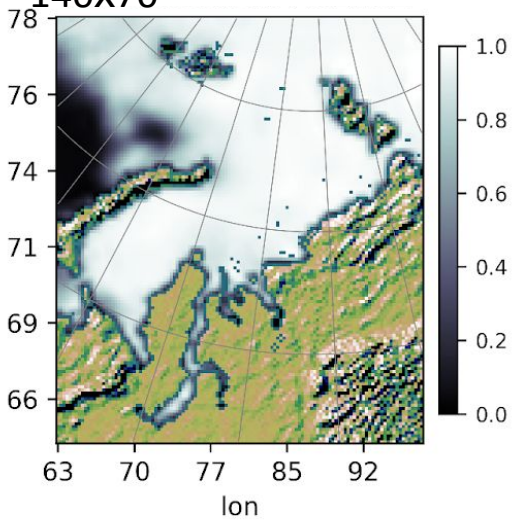
Ожидаемые приложения

1) **Работает в одну строку для любого разрешения изображений**

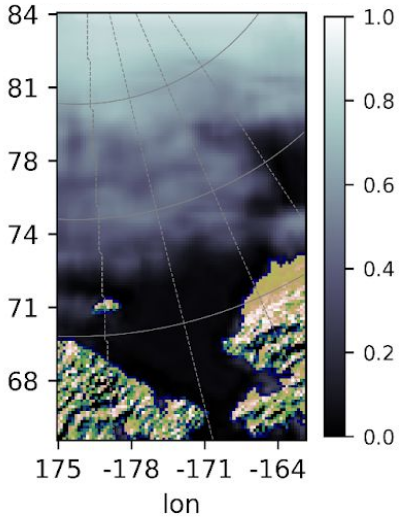
2) **Работает в одну строку для любого числа входных каналов (заблаговременности прогноза)**

<u>Задача предсказания концентрации льда на срок от 2х месяцев до 1 года</u>

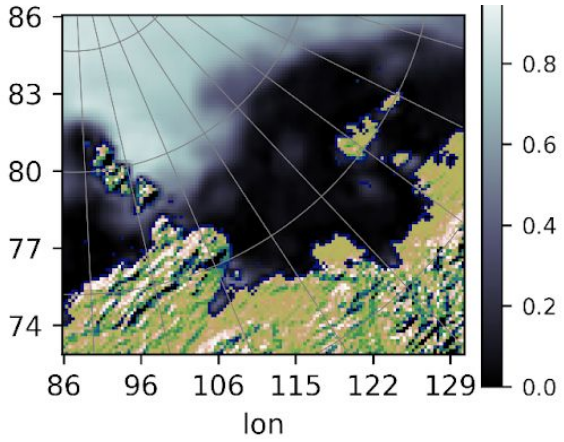
Карское море – 140x70



Чукотское море – 85x145



Море Лаптевых – 110x130



Менее ожидаемые приложения

Модель TorchCNNBuilder'а в роли ансамблирующей модели для нейросетевого ансамбля

Ограничение:
Все элементы ансамбля имеют одинаковое разрешение и выступают в роли каналов входной картинки

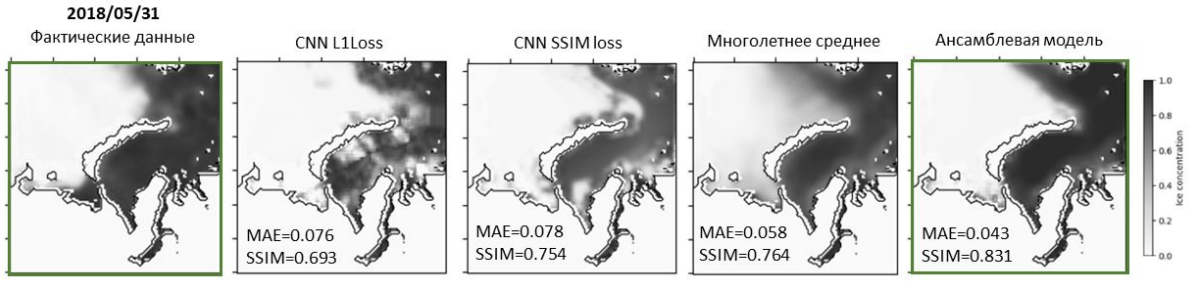
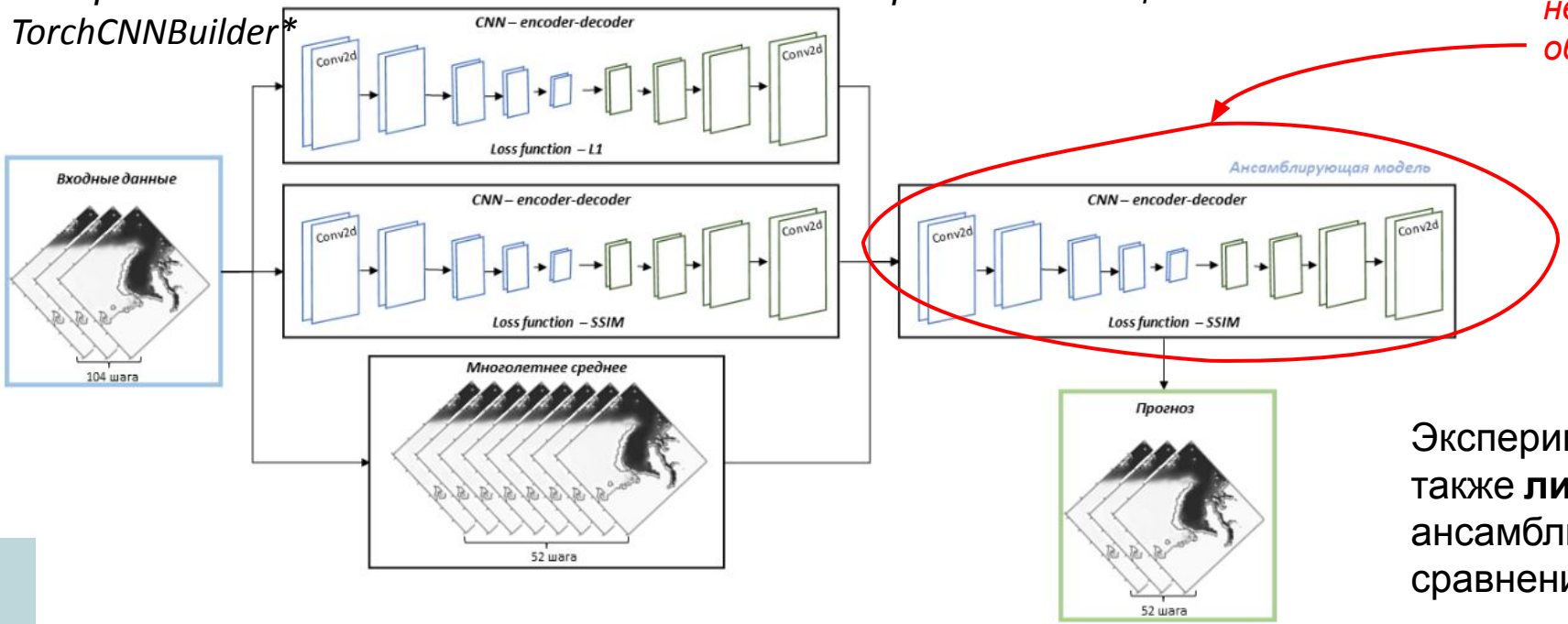


Схема ансамблевой модели для пространственно-временного прогнозирования
нейросетевые элементы ансамбля также собраны с помощью TorchCNNBuilder



сверточная сеть в роли нелинейной объединяющей функции

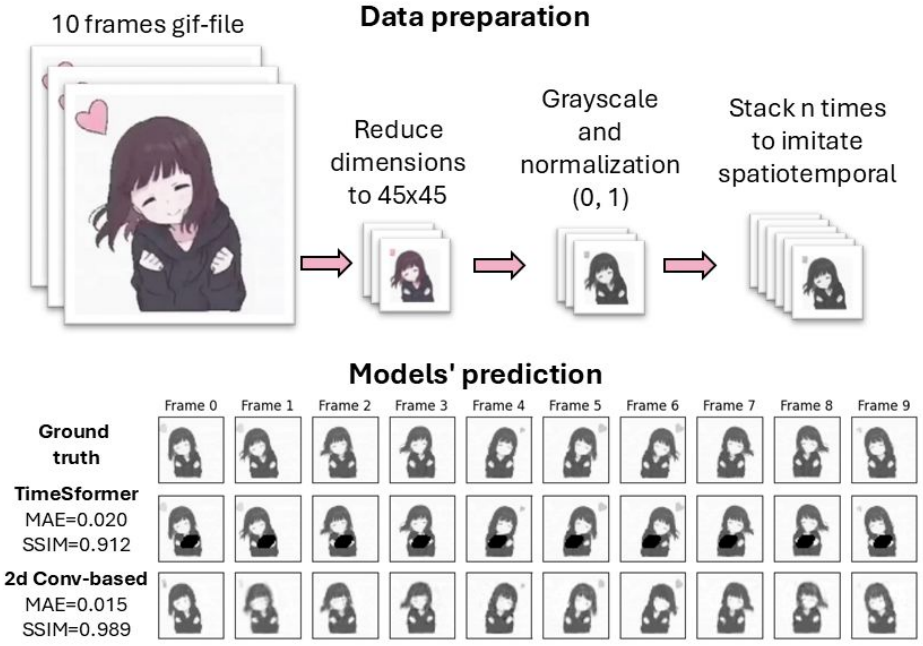
```
model = ForecasterBase(
                input_size=[125, 125],
                in_time_points=3,
                out_time_points=1,
                n_layers=5)
```

Эксперименты с **полносвязными** сетями, а также **линейными** моделями в роли ансамблирующих дали **худшее** качество в сравнении с неглубокой сверточной сетью

Менее ожидаемые приложения

Бейзлайн для задачи предсказания видео

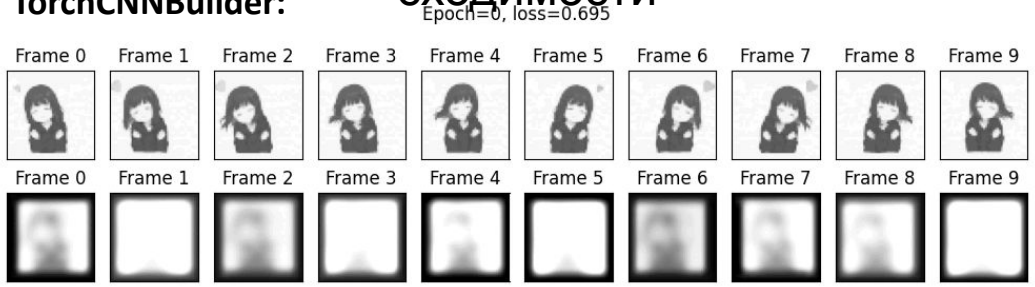
Циклическое видео – как аналог естественного процесса без шумовой компоненты, синтетические данные



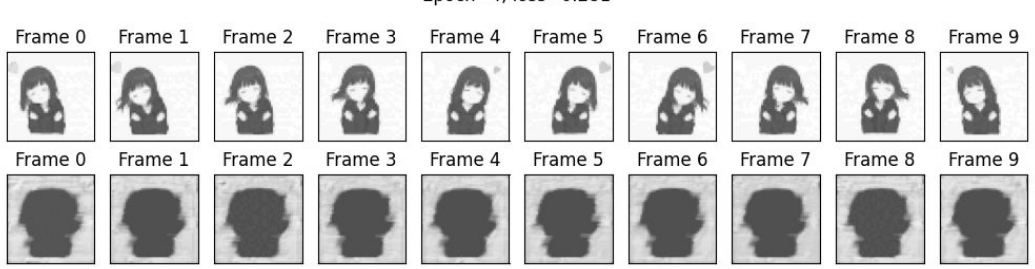

```
train_dataset = multi_output_tensor(data=train,
                                    pre_history_len=20,
                                    forecast_len=10)

model = ForecasterBase(input_size=[45, 45],
                       in_time_points=20,
                       out_time_points=10,
                       n_layers=5,
                       finish_activation_function=nn.ReLU(inplace=True))
```

Оценка процессов сходимости

TorchCNNBuilder:



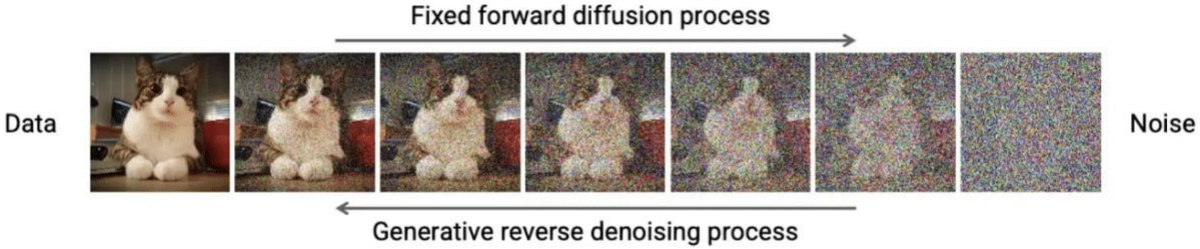
TimeSFormer:



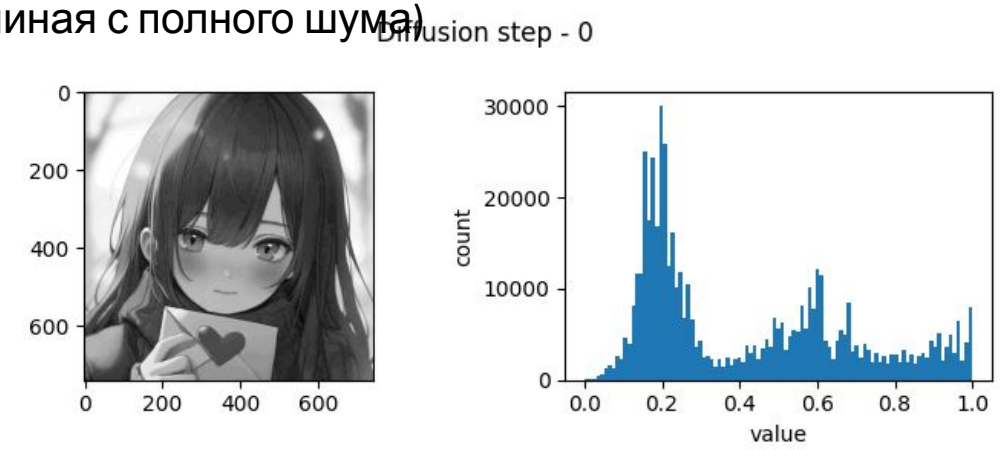
Внезапные приложения (1/2)

Модель TorchCNNBuilder'а предсказывает шум как элемент диффузионной сети

1) Прямая диффузия: формируется обучающая выборка из картинок
2) Данные: зашумленные картинки на разных шагах диффузии, нанесенная компонента шума, номер шага диффузии
3) Модель: по зашумленной картинке учится предсказывать компоненту шума на этой картинке на определенном шаге
4) Обратная диффузия: модель для каждого шага диффузии предсказывает «расшумляющую» компоненту (начиная с полного шума)



Неглубокая сверточная сеть в роли бейзлайна



Прямой диффузионный процесс – изменения исходного распределения картинки шагами, пока не прийдем к полному шуму

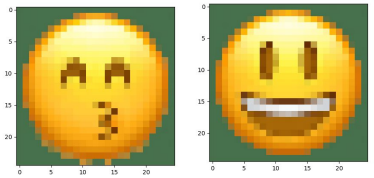
- В качестве данной модели обычно используется модифицированный U-Net с механизмом внимания.
- Среди недостатков - нужна модель под конкретную задачу, а написать код для процессов диффузии хочется гибким под любые данные.
- Проверять банальную работоспособность проще на человекочитаемых картинках.

Внезапные приложения (2/2)

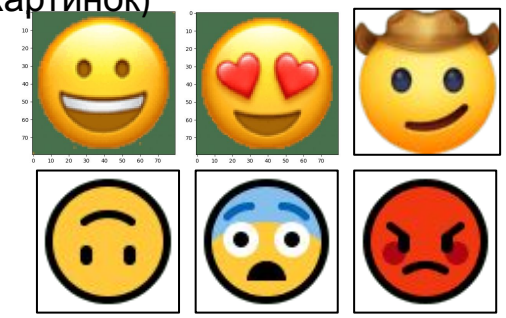
Модель TorchCNNBuilder'а предсказывает шум как элемент диффузионной сети

Ресайз картинки до 25x25

d = 25



Обучающий датасет (600 картинок)

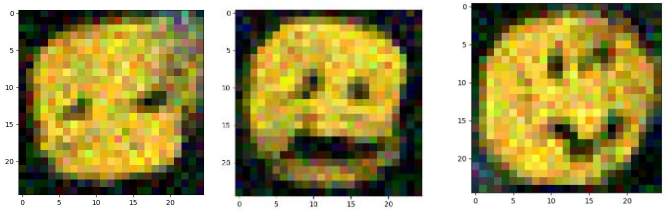


Ресайз картинки до 70x70

d = 70

Гибкость инструмента позволяет ставить много экспериментов без копания внутри модели

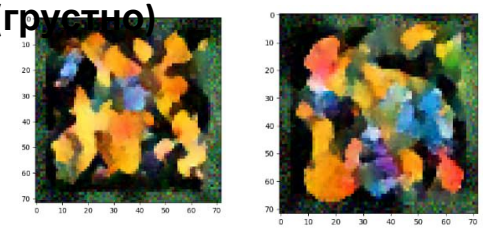
Результат:

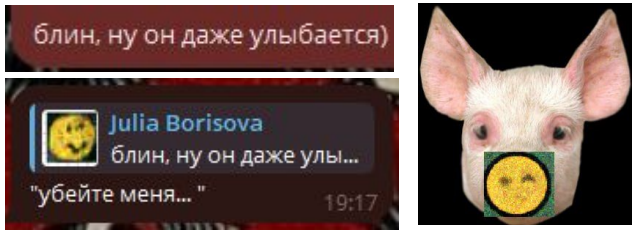


Минимальная модель

```
model = ForecasterBase(
                input_size=(d, d),
                n_layers=5,
                in_time_points=4,
                out_time_points=3)
```

Результат: (грустно)

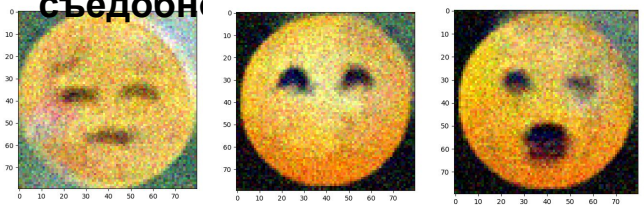





```
model = ForecasterBase(
                input_size=(70, 70),
                n_layers=5,
                in_time_points=4,
                out_time_points=3,
                convolve_params={'kernel_size': (15, 15)},
                transpose_convolve_params={'kernel_size': (15, 15)}
                )
```

Прокинуто до слоев PyTorch

Результат: (грустно, но съедобн



Развитие – интересные задачи

1) **Приложения, ограничения применимости** – для каких данных простые свертки не работают?
2) **NAS** – что если натюнить базовую конфигурацию?
3) **Трехмерные свертки для трехмерных данных** – как правильно выбрать ядро для 3d?
4) **U-net в три строки** – техническая реализация skip connection’ов
5) **Свернули – *сделали что-то еще* – развернули**: техническая реализация точки доступа к энкодеру и декодеру

https://github.com/ChrisLisbon/TorchCNNBuilder

Борисова Юлия
https://t.me/ChrisLisbon

@CHRISLISBON

🔍 - исследовательские

🛠 - технические

Спасибо за внимание!

ITMO more than a UNIVERSITY

NSS Lab

Natural Systems Simulation Lab of ITMO University

Новостной канал tg

https://t.me/NSS_group

Команда, публикации

https://itmo-nss-team.github.io/

GitHub, проекты

https://github.com/ITMO-NSS-team