

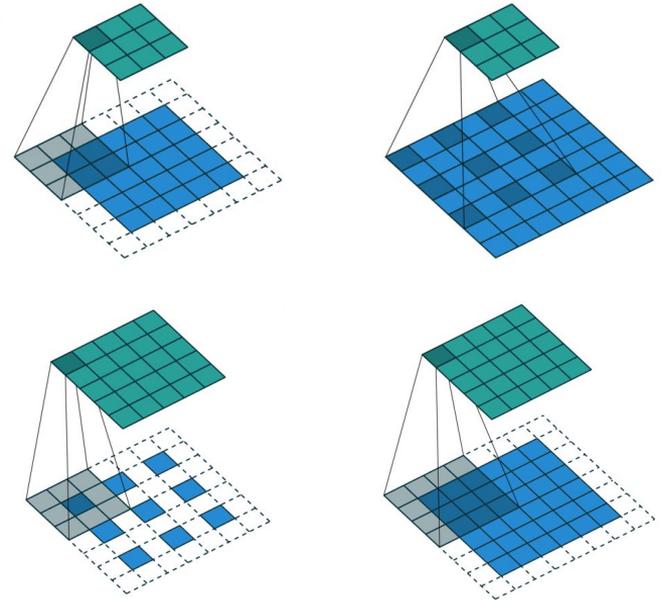


Как писать CNN-модели и не утонуть в свертках

Кузнецов Андрей
ITMO University, Saint-Petersburg
лаборант LISA
2024 г.

Проблема и мотивация

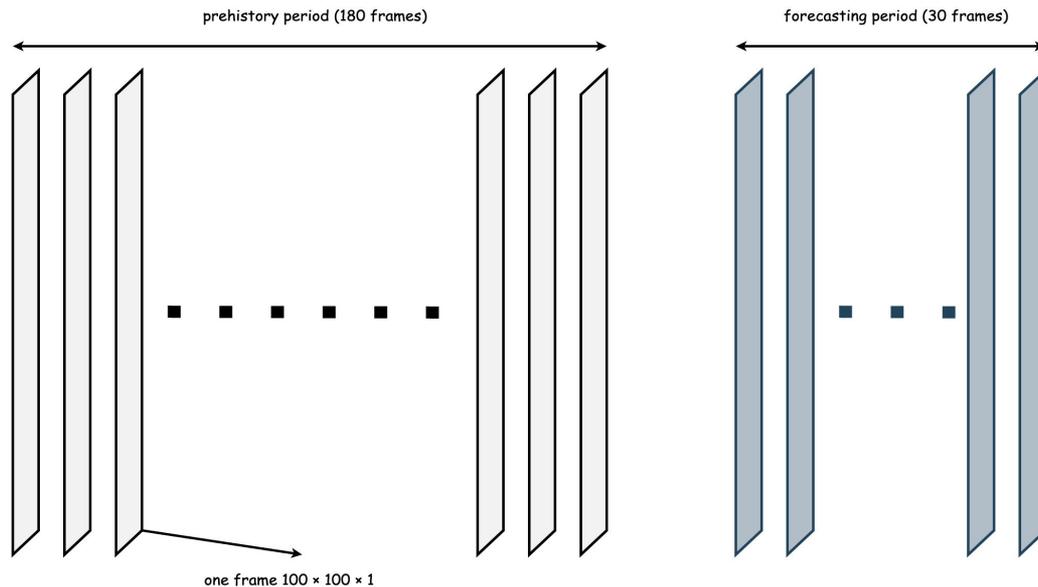
- сложность контролирования размерностей тензора в процессе прохода через сверточные слои
- неудобство корректной подборки гиперпараметров сверточных слоев
- желание проведения большого количества экспериментов с разными слоями и параметрами в разумных временных рамках



Примеры сверточных операций
[https://github.com/vdumoulin/conv_arithmetic]

Предобработка временных рядов

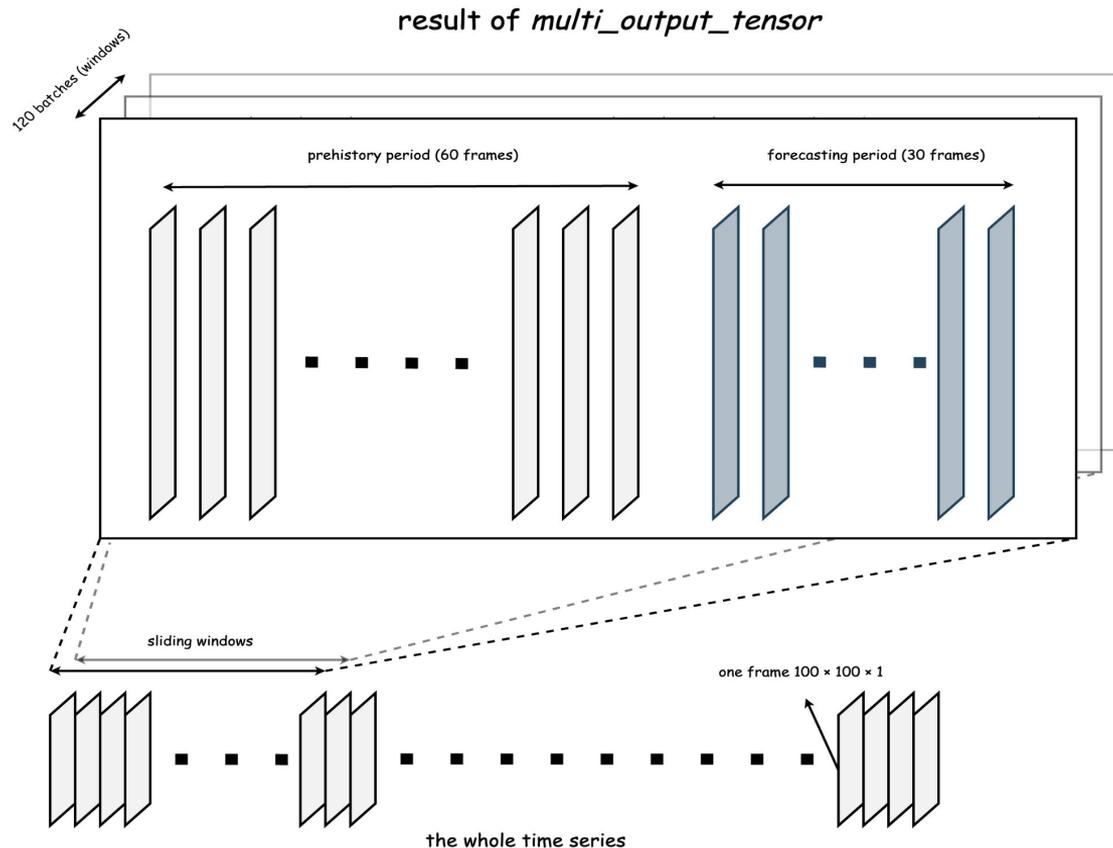
result of *single_output_tensor*



Тензоры могут быть любой размерности (в примере они двумерные)

Предобработка временных рядов

Тензоры могут быть любой размерности (в примере они двумерные)



Предобработка временных рядов

```
dataset = multi_output_tensor(data=noise_data,
                              additional_x=[noise_data.copy(), noise_data.copy(), noise_data.copy()],
                              additional_is_array=True,
                              forecast_len=30,
                              pre_history_len=60,
                              threshold=0.5,
                              x_binarize=True)

for i, batch in enumerate(dataset):
    print(f'batch number: {i}',
          f'new stacked X shape: {batch[0].shape}\nY shape: {batch[1].shape}',
          f'new stacked X max: {batch[0].max()} | min: {batch[0].min()}\nY max: {batch[1].max()} | min: {batch[1].min()}',
          sep='\n',
          end='\n\n')
    if i == 1:
        break
print(f'Dataset len (number of batches/X-windows): {len(dataset)}')
```

доп.временные ряды

предыстория и дальность прогноза

порог бинаризации

бинаризация X

```
batch number: 0
new stacked X shape: torch.Size([60, 4, 100, 100])
Y shape: torch.Size([30, 100, 100])
new stacked X max: 1.0 | min: 0.0
Y max: 1.0 | min: 0.0

batch number: 1
new stacked X shape: torch.Size([60, 4, 100, 100])
Y shape: torch.Size([30, 100, 100])
new stacked X max: 1.0 | min: 0.0
Y max: 1.0 | min: 0.0

Dataset len (number of batches/X-windows): 120
```

проверка размерностей данных

Примеры API builder

```
In [2]: from torchcnnbuilder.builder import conv1d_out, conv2d_out, conv3d_out
```

импорт функций
для подсчета
результатов
после сверток

$$H_{out} = \lfloor \frac{H_{in} + 2 \times padding[0] - dilation[0] \times (kernel[0] - 1) + 1}{stride[0]} \rfloor + 1$$

```
In [3]: new_size = conv1d_out(input_size=33,
                             kernel_size=5,
                             stride=2)

print(f'Tensor size after nn.Conv1d: {new_size}')
```

параметры
свертки

Tensor size after nn.Conv1d: 15

Аналогично для транспонированных
операций

Класс Builder

```
In [8]: from torchcnnbuilder.builder import Builder
```

Initialization params:

- **input_size** (Sequence[int]): input size of the input tensor
- **minimum_feature_map_size** (Union[Sequence[int], int]): minimum feature map size. Default: 5
- **max_channels** (int): maximum number of layers after any convolution. Default: 512
- **min_channels** (int): minimum number of layers after any convolution. Default: 32
- **activation_function** (nn.Module): activation function. Default: nn.ReLU(inplace=True)
- **finish_activation_function** (Union[str, Optional[nn.Module]]): last activation function, can be same as activation_function (str 'same'). Default: None
- **default_convolve_params** (dict[str, Union[int, tuple]]): parameters of convolutional layers (by default same as in torch)
- **default_transpose_params** (dict[str, Union[int, tuple]]): parameters of transpose convolutional layers (by default same as in torch)

Other attributes:

- **conv_channels** (List[int]): list of output channels after each convolutional layer
- **transpose_conv_channels** (List[int]): list of output channels after each transposed convolutional layer
- **conv_layers** (List[tuple]): list of output tensor sizes after each convolutional layer
- **transpose_conv_layers** (List[tuple]): list of output tensor sizes after each transposed convolutional layer

Suppose we create a CNN-model for a tensor of size 100 by 100, let the smallest feature map after the convolutions be of size 3 by 3

```
In [9]: builder = Builder(input_size=[125, 125],  
                        minimum_feature_map_size=3)
```

Описание атрибутов классов

Подробное описание каждого метода есть в репозитории проекта в директории examples

Класс Builder

описание метода
ascending



$$\left\{ \begin{array}{ll} \text{channel}_i = \text{start} \times \text{ratio}^i, & i = 1 \dots n & \text{if } \text{ascending} = \text{False} \\ \text{stop} = \lfloor \frac{(\text{input_size}[0] + \text{input_size}[1]) * 0.5}{2} \rfloor + \text{in_channels} & & \text{if } \text{ascending} = \text{True} \\ \text{step} = \frac{\text{stop} - \text{in_channels}}{n_layers} & & \\ \text{channels} = \text{range}(\text{in_channels}, \text{stop}, \text{step}) & & \end{array} \right.$$

```
builder.build_convolve_sequence(n_layers=2,
                                in_channels=3,
                                params={'kernel_size': 9},
                                sub_blocks=2,
                                ascending=True)
```

параметры сверток

Из документации метода build_convolve_sequence

```
builder.build_convolve_sequence(n_layers=3,
                                in_channels=3,
                                ascending=False, # by default
                                ratio=3)
```

```
Sequential(
  (conv 1): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU(inplace=True)
  )
  (conv 2): Sequential(
    (0): Conv2d(32, 96, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU(inplace=True)
  )
  (conv 3): Sequential(
    (0): Conv2d(96, 288, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU(inplace=True)
  )
)
```

```
Sequential(
  (conv 1): Sequential(
    (sub-block 1): Sequential(
      (0): Conv2d(3, 34, kernel_size=(9, 9), stride=(1, 1), padding=(4, 4))
      (1): ReLU(inplace=True)
    )
    (sub-block 2): Sequential(
      (0): Conv2d(34, 34, kernel_size=(9, 9), stride=(1, 1), padding=(4, 4))
      (1): ReLU(inplace=True)
    )
  )
  (conv 2): Sequential(
    (sub-block 1): Sequential(
      (0): Conv2d(34, 65, kernel_size=(9, 9), stride=(1, 1), padding=(4, 4))
      (1): ReLU(inplace=True)
    )
    (sub-block 2): Sequential(
      (0): Conv2d(65, 65, kernel_size=(9, 9), stride=(1, 1), padding=(4, 4))
      (1): ReLU(inplace=True)
    )
  )
)
```

сверточная последовательность

сверточная последовательность

Класс Builder

```
: builder.finish_activation_function = nn.Softmax()
deconv_layer = builder.build_transpose_convolve_block(in_channels=3,
                                                       out_channels=64,
                                                       normalization='dropout',
                                                       p=0.2,
                                                       sub_blocks=3,
                                                       last_block=True)

deconv_layer

: Sequential(
  (transpose sub-block 1): Sequential(
    (0): ConvTranspose2d(3, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): Dropout2d(p=0.2, inplace=False)
    (2): ReLU(inplace=True)
  )
  (transpose sub-block 2): Sequential(
    (0): ConvTranspose2d(3, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): Dropout2d(p=0.2, inplace=False)
    (2): ReLU(inplace=True)
  )
  (transpose sub-block 3): Sequential(
    (0): ConvTranspose2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): Dropout2d(p=0.2, inplace=False)
    (2): Softmax(dim=None)
  )
)
```

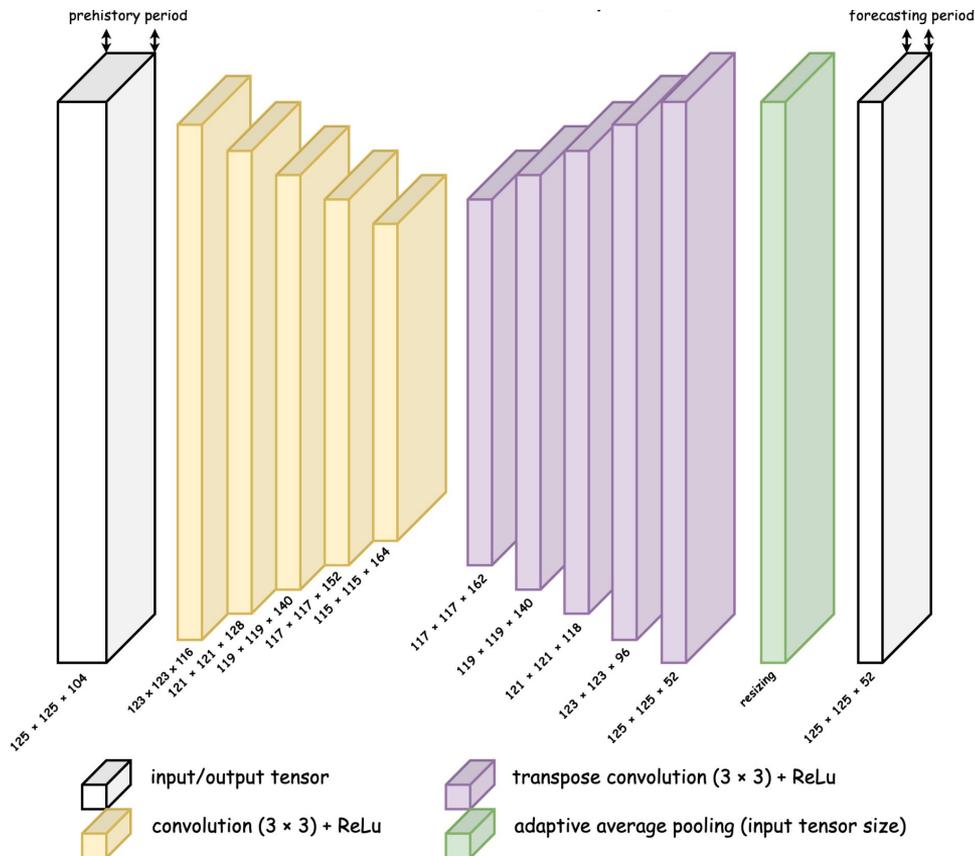
замена финальной функции активации

настройка нормализации

сверточный блок

Можно строить сверточные блоки отдельно (на примере транспонированной свертки)

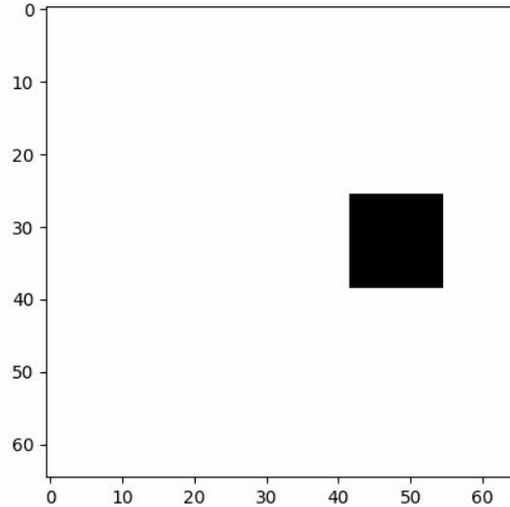
Шаблоны моделей (ForecasterBase)



Модель была предложена для решения задачи прогнозирования концентрации льда в статье “Forecasting of Sea Ice Concentration using CNN, PDE discovery and Bayesian Networks”

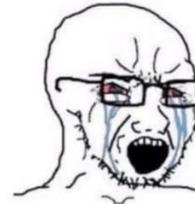
[<https://www.sciencedirect.com/science/article/pii/S1877050923020094>]

Возможности архитектуры



train часть 2-мерного временного ряда
(движение квадрата по кругу)

Kaggle



Нам бы очень этого не хотелось, но мы вынуждены ограничить ресурсы GPU до 30 часов в неделю (((((

Colab



10 минут пообучал, теперь плати деньги.

```
In [3]: from torchcnnbuilder.models import ForecasterBase
```

```
In [4]: import torch.nn as nn
```

```
# example of initialization
```

```
model = ForecasterBase(input_size=[65, 65],  
                        in_channels=120,  
                        out_channels=40,  
                        n_layers=5,  
                        normalization='batchnorm',  
                        finish_activation_function=nn.ReLU(inplace=True))
```

входной размер матрицы
предыстория
дальность прогноза
кол-во слоев
добавление нормализации
финальная ф-ция активации

Параметры инициализации

```
In [11]: train_dataset = multi_output_tensor(data=train,  
                                             pre_history_len=120,  
                                             forecast_len=40)  
  
test_dataset = single_output_tensor(data=test,  
                                    forecast_len=40)
```

предыстория и дальность прогноза

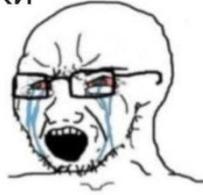
Подготовка датасета

Немного мемов про мой процесс обучения

C++ разработчики



Visual studio лучшая среда
для разработки!!!



Нет, Rider лучше!!!

ML-разработчики



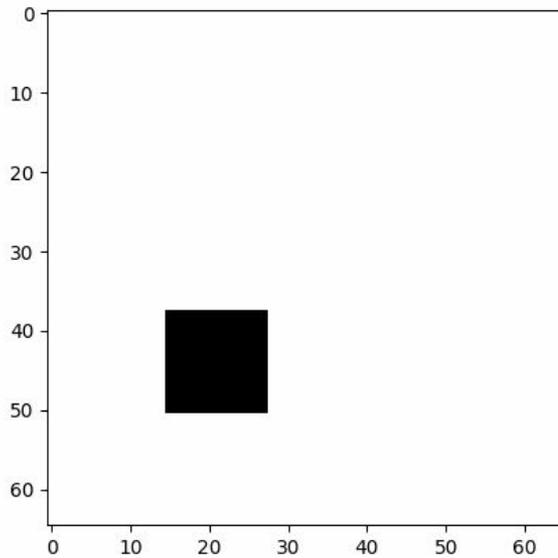
Я пишу и компилирую код
в браузере



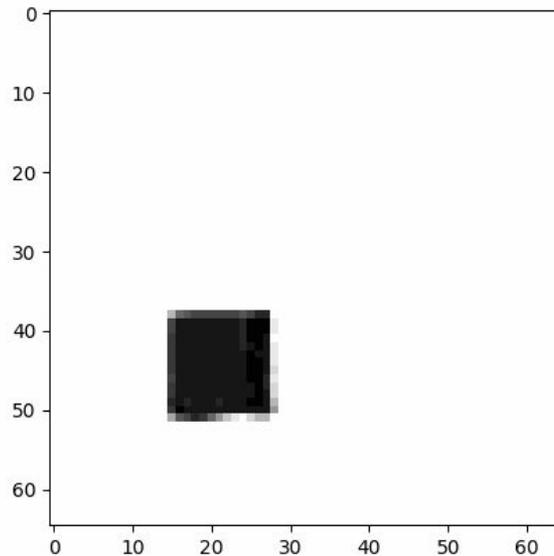
Я тоже



Результат



test



predict

main 2 Branches 0 Tags

Go to file Add file Code

kdduha	updating gitignore	843273d · 3 days ago	46 Commits
examples	update examples		3 days ago
torchcnbuilder	minor fixes		3 days ago
.gitignore	updating gitignore		3 days ago
LICENSE	minor fxirs		last week
README.md	minor fixes		3 days ago
requirements.txt	pep8 fixes		4 days ago
setup.cfg	initial commit		last week
setup.py	minor fixes		3 days ago

README BSD-3-Clause license

TorchCNNBuilder

python 3.8 | 3.9 | 3.10 pypi package 0.0.17

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 torchcnbuilder
```

About

Framework for the automatic creation of CNN architectures

open-source time-series cv torch
cnn-architecture

- Readme
- BSD-3-Clause license
- Activity
- Custom properties
- 4 stars
- 4 watching
- 0 forks

Report repository

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Contributors (2)

- kdduha Andrey Kuznetsov
- ChrisLisbon Julia Borisova

Languages

- Python 100.0%

Suggested workflows

Based on your tech stack

ИТМО

Подробное описание проекта с примерами всей функциональности и документацией



GitHub страница разработанной библиотек

Спасибо за внимание!

iTMO *re than a*
UNIVERSITY