ФКН

Образовательная программа
«Программная инженерия»

Научно-исследовательский семинар
«Технологии компьютерного зрения»

# Производительность нейронных сетей. Развертывание в C++. OpenCV, JIT, ONNX

**Выполнил**:
студент группы БПИ213
Абрамов Александр Сергеевич

Higher School of Economics
Faculty of Computer science

# Зачем?

## Python

✅ Легкость изучения и удобство разработки

❌ Низкая скорость работы, высокое использование ресурсов и памяти

❌ Сложность масштабирования, низкая стабильность

## C++

❌ Высокая сложность изучения и написания кода

✅ Высокая производительность, низкое потребление ресурсов

✅ Стабильность, относительная простота масштабирования

# OpenCV, Трансформации

OpenCV - библиотека с открытым исходным кодом, реализующая алгоритмы компьютерного зрения и обработки изображений. Разработана российской компанией Itseez; в мае 2016 года была приобретена корпорацией Intel. https://opencv.org/

### imread()

```
Mat cv::imread ( const String &  filename,
                 int             flags = IMREAD_COLOR
               )
```

**Python:**

```
cv.imread( filename[, flags] ) -> retval
```

### cvtColor()

```
void cv::cvtColor ( InputArray   src,
                    OutputArray  dst,
                    int          code,
                    int          dstCn = 0
                  )
```

### transposeND()

```
void cv::transposeND ( InputArray                src,
                       const std::vector< int > & order,
                       OutputArray               dst
                     )
```

## Image ROI

We use the function: **roi (rect)**

**Parameters**

**rect** rectangle Region of Interest.

### convertTo()

```
void cv::Mat::convertTo ( OutputArray  m,
                          int          rtype,
                          double       alpha = 1 ,
                          double       beta = 0
                        )              const
```

### forEach() [1/2]

```
template<typename _Tp , typename Functor >
void cv::Mat::forEach ( const Functor &  operation )
```

Higher School of Economics
Faculty of Computer science

# Resize: Pillow vs OpenCV

◆ resize()

```
void cv::resize ( InputArray  src,
                  OutputArray dst,
                  Size        dsize,
                  double      fx = 0 ,
                  double      fy = 0 ,
                  int         interpolation = INTER_LINEAR
                )
```

OpenCV не реализует сглаживание, а в Pillow его нельзя отключить!

Решение: pilllow_resize

Подробнее: статья

RESIZE

CLASS torchvision.transforms.v2.Resize(*size: Union[int, Sequence[int]], interpolation: Union[InterpolationMode, int] = InterpolationMode.BILINEAR, max_size: Optional[int] = None, antialias: Optional[bool] = True*) [SOURCE]

Resize the input to the given size.

- **antialias** (*bool, optional*) –

  Whether to apply antialiasing. It only affects **tensors** with bilinear or bicubic modes and it is ignored otherwise: on PIL images, antialiasing is always applied on bilinear or bicubic modes; on other modes (for PIL images and tensors), antialiasing makes no sense and this parameter is ignored. Possible values are:

  - `True` (default): will apply antialiasing for bilinear or bicubic modes. Other mode aren't affected. This is probably what you want to use.
  - `False`: will not apply antialiasing for tensors on any mode. PIL images are still antialiased on bilinear or bicubic modes, because PIL doesn't support no antialias.
  - `None`: equivalent to `False` for tensors and `True` for PIL images. This value exists for legacy reasons and you probably don't want to use it unless you really know what you are doing.

  The default value changed from `None` to `True` in v0.17, for the PIL and Tensor backends to be consistent.

# TorchScript, JIT

## Tracing

- Запускает модель на конкретном вводе, запоминая выполняемые операции
- Для работы требуется пример входных данных модели
- Все объекты, не являющиеся torch.Tensor превращаются в константы
- Не поддерживает ветвление (if, for) на базе входных данных

TORCH.JIT.TRACE

torch.jit.trace(*func*, *example_inputs=None*, *optimize=None*, *check_trace=True*, *check_inputs=None*, *check_tolerance=1e-05*, *strict=True*, *_force_outplace=False*, *_module_class=None*, *_compilation_unit=<torch.jit.CompilationUnit object>*, *example_kwarg_inputs=None*, *_store_inputs=True*) [SOURCE]

Trace a function and return an executable or `ScriptFunction` that will be optimized using just-in-time compilation. Tracing is ideal for code that operates only on `Tensor`s and lists, dictionaries, and tuples of `Tensor`s.

## Scripting

- Использует компилятор TorchScript, являющийся подмножеством Python
- Поддерживает ветвление, а также разрешает не только тензорные операции
- Не реализует многие возможности Python (in-place операции, динамическая типизация, классы)

TORCH.JIT.SCRIPT

torch.jit.script(*obj*, *optimize=None*, *_frames_up=0*, *_rcb=None*, *example_inputs=None*) [SOURCE]

Scripting a function or `nn.Module` will inspect the source code, compile it as TorchScript code using the TorchScript compiler, and return a `ScriptModule` or `ScriptFunction`. TorchScript itself is a subset of the Python language, so not all features in Python work, but we provide enough functionality to compute on tensors and do control-dependent operations. For a complete guide, see the TorchScript Language Reference.

Higher School of Economics
Faculty of Computer science

# Open Neural Network Exchange

ONNX - открытая спецификация для представления моделей глубинного обучения. Исходно разрабатывалась под названием Toffee командой PyTorch в Facebook. В сентябре 2017 г. была переименована и опубликована совместно Facebook и Microsoft. Позже к разработке присоединились IBM, Huawei, Intel, AMD и др. https://onnx.ai/

## TorchScript-based

- Требует предварительной конвертации в TorchScript с помощью JIT
- Имеет те же ограничения использования, что и JIT
- Конвертация в TorchScript производится автоматически с помощью tracing

```
torch.onnx.export(model, args, f, export_params=True, verbose=False, training=
<TrainingMode.EVAL: 0>, input_names=None, output_names=None, operator_export_type=
<OperatorExportTypes.ONNX: 0>, opset_version=None, do_constant_folding=True,
dynamic_axes=None, keep_initializers_as_inputs=None, custom_opsets=None,
export_modules_as_functions=False, autograd_inlining=True) [SOURCE]
```

Exports a model into ONNX format.

## TorchDynamo-based

- Экспериментальная технология
- Интегрируется в компилятор Python и динамически переписывает код в FX-граф

```
torch.onnx.dynamo_export(model, /, *model_args, export_options=None, **model_kwargs)
```

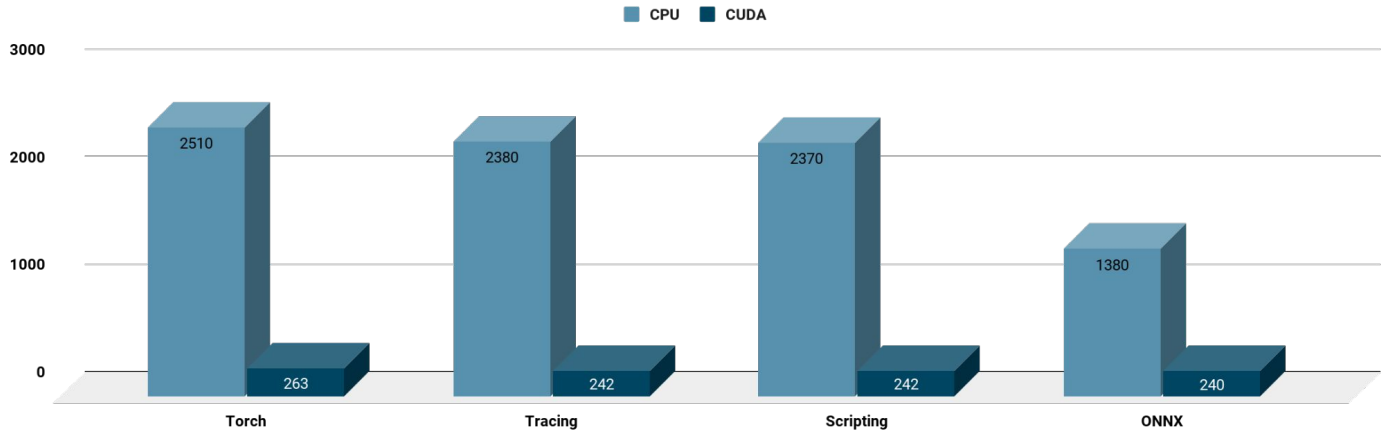
Export a torch.nn.Module to an ONNX graph.

**Parameters**

- **model** (*Union*[*Module*, *Callable*]) – The PyTorch model to be exported to ONNX.
- **model_args** – Positional inputs to `model`.
- **model_kwargs** – Keyword inputs to `model`.
- **export_options** (*Optional*[*ExportOptions*]) – Options to influence the export to ONNX.

# Сравнение

| | Torch | Tracing | Scripting | ONNX |
|---|---|---|---|---|
| **CPU** | 2.51s | 2.38s | 2.37s | 1.38s |
| **CUDA** | 263ms | 242ms | 242ms | 240ms |




* Intel core i5-12600K, 32GB DDR5 RAM, Nvidia GeForce GTX 1660 Super

Higher School of Economics
Faculty of Computer Science