i need to convert yolov5s.pt into onnx format to use it with visual c++ (visual studio 2015, and opencv 4.5.0). The model is trained on python3.7.6 and torch 1.7.1.

ChatGPT said:
To convert your yolov5s.pt model to ONNX format (compatible with OpenCV 4.5.0 and C++), follow these steps:

✅ Requirements

Your environment is:

Python 3.7.6

PyTorch 1.7.1

Model: yolov5s.pt

Target: OpenCV 4.5.0 with Visual C++ (VS2015)

Ensure that:

bash
Copy
Edit

```
pip install torch==1.7.1 torchvision==0.8.2 onnx==1.8.0 onnxruntime
```

🔧 Step-by-step Conversion

1. Clone YOLOv5 repository

Use the same version you trained the model on (assuming it's close to v3.0 for torch 1.7.1):

bash
Copy
Edit

```
git clone https://github.com/ultralytics/yolov5.git
cd yolov5
git checkout v3.0  # Optional: pick the version that matches your model
pip install -r requirements.txt
```

2. Export to ONNX

Run the export.py script provided by YOLOv5:

bash
Copy
Edit

```
python export.py --weights yolov5s.pt --img 640 --batch 1 --device cpu --include onnx
```

This will generate yolov5s.onnx in the same directory.

--img 640 specifies the input image size (make sure it's what your model was trained on).
You can also add --simplify to reduce complexity if OpenCV has trouble with the ONNX.

✅ Verify ONNX with OpenCV 4.5.0
Make sure OpenCV 4.5.0 supports your ONNX opset version. You can check it like this:

python
Copy
Edit

```python
import onnx
model = onnx.load("yolov5s.onnx")
print(model.opset_import)
```

OpenCV 4.5.0 supports ONNX opset up to version 11.

If your model uses a newer opset, you'll need to export with a lower opset:

bash
Copy
Edit

```bash
python export.py --weights yolov5s.pt --img 640 --batch 1 --device cpu --include onnx --opset 11
```

📦 Load in C++ with OpenCV
Sample C++ code:

cpp
Copy
Edit

```cpp
cv::dnn::Net net = cv::dnn::readNetFromONNX("yolov5s.onnx");

// Set preferable backend and target
net.setPreferableBackend(cv::dnn::DNN_BACKEND_OPENCV);
net.setPreferableTarget(cv::dnn::DNN_TARGET_CPU);

// Prepare blob from image
cv::Mat img = cv::imread("image.jpg");
cv::Mat blob;
cv::dnn::blobFromImage(img, blob, 1.0 / 255.0, cv::Size(640, 640), cv::Scalar(), true, false);

net.setInput(blob);
cv::Mat output = net.forward();  // For YOLOv5, this may need postprocessing
```

🧠 Notes for Compatibility with OpenCV
NMS and Post-processing: YOLOv5's ONNX output requires additional postprocessing for confidence filtering, NMS, and coordinate rescaling. You must handle this manually in C++.

ONNX Simplification: If OpenCV fails to load the model:

Try --simplify during export.

Or simplify with onnx-simplifier:

bash
Copy
Edit

```
pip install onnx-simplifier
python -m onnxsim yolov5s.onnx yolov5s-simplified.onnx
```

If you'd like, I can help you generate the exact C++ code for postprocessing the YOLOv5 ONNX output. Would you like that?

You said:
i need to convert yolov5s.pt into onnx format to use it with visual c++ (visual studio 2015, and opencv 4.5.0). The model is trained on python3.7.6 and torch 1.8.0.

ChatGPT said:
Thanks for the update. With Python 3.7.6 and PyTorch 1.8.0, you're in a good position to convert yolov5s.pt to ONNX format for Visual Studio 2015 + OpenCV 4.5.0.

Here's a tailored step-by-step guide based on your environment:

✅ 1. Set Up the Environment

Install the required libraries:

bash
Copy
Edit

```
pip install torch==1.8.0 torchvision==0.9.0 onnx==1.9.0 onnxruntime
```

✅ 2. Clone YOLOv5 (Compatible Version)

Since you're using torch 1.8.0, use YOLOv5 v4.0 (released around the same time):

bash
Copy
Edit

```
git clone https://github.com/ultralytics/yolov5.git
cd yolov5
git checkout v4.0
pip install -r requirements.txt
```

✅ 3. Convert yolov5s.pt to ONNX

Run the export script:

bash
Copy
Edit

```
python export.py --weights yolov5s.pt --img 640 --batch 1 --device cpu --include onnx --opset 11
```

Important:

--opset 11: Required because OpenCV 4.5.0 only supports ONNX opset ≤ 11.

--img 640: Must match the size your model expects (default YOLOv5 is 640).

Output will be yolov5s.onnx.

✅ 4. (Optional) Simplify the ONNX Model

This can help if OpenCV fails to load the model:

bash
Copy
Edit

```
pip install onnx-simplifier
python -m onnxsim yolov5s.onnx yolov5s-simplified.onnx
```

Use yolov5s-simplified.onnx in your C++ project.

✅ 5. Load YOLOv5 ONNX in Visual C++ (OpenCV 4.5.0)

Here’s a minimal C++ example using OpenCV DNN:

cpp
Copy
Edit

```
#include <opencv2/opencv.hpp>
#include <opencv2/dnn.hpp>

int main() {
    // Load model
    cv::dnn::Net net = cv::dnn::readNetFromONNX("yolov5s.onnx");

    // Optional: Set backend and target
    net.setPreferableBackend(cv::dnn::DNN_BACKEND_OPENCV);
    net.setPreferableTarget(cv::dnn::DNN_TARGET_CPU);

    // Load image
    cv::Mat img = cv::imread("image.jpg");
    cv::Mat blob;

    // Preprocess
    cv::dnn::blobFromImage(img, blob, 1/255.0, cv::Size(640, 640), cv::Scalar(), true, false);
    net.setInput(blob);

    // Inference
    std::vector<cv::Mat> outputs;
    net.forward(outputs, net.getUnconnectedOutLayersNames());

    // You must post-process YOLOv5 output manually (filter boxes, NMS, etc.)
    // I can help write that if you need.

    return 0;
}
```

⚠️ Known OpenCV 4.5.0 Limitations

OpenCV DNN does not include YOLO post-processing.

You need to:

Decode the output tensor

Apply confidence threshold

Apply non-max suppression (NMS)

Rescale boxes to original image size

📦 Want Help with C++ Post-processing?

Let me know and I can provide a clean postprocessing C++ function for YOLOv5 ONNX output (based on OpenCV's format).

Would you like that?

You said:

I changed the directory into yolov5/models because export.py was there. And I command this.

```
python export.py --weights yolov5s.pt --img 640 --batch 1 --device cpu --include onnx
```

And I got this error.

```
    from utils.google_utils import attempt_download
ModuleNotFoundError: No module named 'utils'
```

ChatGPT said:

Ah, good catch. That error is happening because you ran export.py from the wrong directory — the script uses relative imports, so it expects to be run from the root of the YOLOv5 repo, not from inside models/.

✅ Fix: Run from YOLOv5 root directory

Instead of this:

bash
Copy
Edit

```
cd yolov5/models
python export.py --weights yolov5s.pt ...
```

Do this:

bash
Copy
Edit

```
cd yolov5
python export.py --weights yolov5s.pt --img 640 --batch 1 --device cpu --include onnx --opset 11
```

The export.py script uses imports like from utils.google_utils import attempt_download, which will fail if you run it from a subdirectory like models/.

🧠 Bonus Tips

Make sure yolov5s.pt is either in the root yolov5/ directory or you provide the full path to it.

You can also add --simplify to help compatibility with OpenCV if needed.

Let me know if you want help verifying the ONNX output or doing inference in C++.

You said:
python: can't open file 'export.py': [Errno 2] No such file or directory

ChatGPT said:
That error means you're not in the right directory when trying to run export.py.

Let’s fix this step-by-step:

✅ Correct Steps
Go to your cloned YOLOv5 directory

Assuming you cloned YOLOv5 like this:

bash
Copy
Edit
```
git clone https://github.com/ultralytics/yolov5.git
```
Then go to the correct directory:

bash
Copy
Edit
```
cd yolov5
```
Now list the files:

bash
Copy
Edit
```
dir  # on Windows
# or
ls   # on Linux/macOS
```
You should see something like:

cpp
Copy
Edit
```
export.py  detect.py  train.py  models/  utils/  data/  yolov5s.pt
```
If export.py is visible in this directory, you are now in the correct location.

Run the export command from this root directory:

bash
Copy

Edit

```
python export.py --weights yolov5s.pt --img 640 --batch 1 --device cpu --include onnx --opset 11
```

🧠 Recap

Don't cd into models/ or utils/

Run export.py from the root yolov5/ folder

If yolov5s.pt is in a different folder, pass the full path to --weights

Let me know if you’d like help verifying the exported ONNX model or writing a C++ inference example.

You said:

```
Namespace(batch_size=1, img_size=[640, 640], weights='yolov5s.pt')
Traceback (most recent call last):
  File "export.py", line 29, in <module>
    model = torch.load(opt.weights, map_location=torch.device('cpu'))['model'].float()
  File "C:\ProgramData\anaconda3\envs\a\lib\site-packages\torch\serialization.py", line 594, in
load
    return _load(opened_zipfile, map_location, pickle_module, **pickle_load_args)
  File "C:\ProgramData\anaconda3\envs\a\lib\site-packages\torch\serialization.py", line 853, in
_load
    result = unpickler.load()
AttributeError: Can't get attribute 'C3' on <module 'models.common' from
'C:\\Users\\Administrator\\yolov5\\models\\common.py'>
```

ChatGPT said:

You're very close! This error means:

The model (yolov5s.pt) was trained using a version of YOLOv5 that defines a module (like C3) differently than what the current models/common.py expects.

This is a common version mismatch problem between:

The version of the model you trained (yolov5s.pt)

The version of the YOLOv5 code you're now using to load/export

✅ Fix: Match YOLOv5 version to your model's version

Step-by-step fix:

Since you trained the model using PyTorch 1.8.0, the likely YOLOv5 version is around v5.0, but the C3 module suggests you need a version at least v5.0 or newer.

1. Try checking out YOLOv5 v5.0

In your yolov5/ directory:

bash
Copy
Edit

```
git fetch
git checkout v5.0
```

Then try exporting again:

bash
Copy
Edit

```
python export.py --weights yolov5s.pt --img 640 --batch 1 --device cpu --include onnx --opset 11
```

🧪 Still not working?

If you're unsure what version of YOLOv5 your .pt file came from, you can try auto-detecting it.

Optionally: Print model keys

You can inspect the model file like this:

python
Copy
Edit

```
import torch
model = torch.load('yolov5s.pt', map_location='cpu')
print(model.keys())
```

If it shows something like 'model': nn.Module(...), but fails with C3, you definitely need the same YOLOv5 version it was trained with.

✅ Summary

Problem	Solution
AttributeError: Can't get attribute 'C3'	Use the same YOLOv5 version the .pt model was trained with
You don't know the version?	Try git checkout v5.0, v5.0.1, v6.0, etc., until it works (usually ≤ v6.0 if torch is 1.8.0)

Let me know if you'd like me to help you identify the YOLOv5 version more precisely from your .pt file.

You said:

```
  File "export.py", line 38, in <module>
    model = attempt_load(opt.weights, map_location=device)  # load FP32 model
  File "C:\Users\Administrator\yolov5\models\experimental.py", line 118, in attempt_load
    ckpt = torch.load(w, map_location=map_location)  # load
  File "C:\ProgramData\anaconda3\envs\a\lib\site-packages\torch\serialization.py", line 594, in load
    return _load(opened_zipfile, map_location, pickle_module, **pickle_load_args)
  File "C:\ProgramData\anaconda3\envs\a\lib\site-packages\torch\serialization.py", line 853, in _load
    result = unpickler.load()
AttributeError: Can't get attribute 'SPPF' on <module 'models.common' from 'C:\\Users\\Administrator\\yolov5\\models\\common.py'>
```

ChatGPT said:

You're still encountering the same type of issue — now with SPPF, instead of C3. This confirms that the YOLOv5 version you're using is too old compared to the version your yolov5s.pt model was trained with.

✅ Solution: Match YOLOv5 repo to your model's version

Your .pt file was trained with a version of YOLOv5 that includes the SPPF module — this was introduced in YOLOv5 v6.0 and later.

🔄 Fix It

Run the following commands in your yolov5 directory to update the repo to the correct version:

bash
Copy
Edit

```
cd yolov5
git fetch
git checkout v6.0
```

Then try the export again:

bash
Copy

Edit

```
python export.py --weights yolov5s.pt --img 640 --batch 1 --device cpu --include onnx --opset 11
```

If you're still getting SPPF-related errors, try going to YOLOv5 v6.2 instead:

bash
Copy
Edit

```
git checkout v6.2
```

🧠 Why This Happens

PyTorch tries to unpickle the model file (.pt), and it needs to find all the exact class definitions (like C3, SPPF) in models/common.py.

If you're using an older or mismatched version of the YOLOv5 code, those definitions won’t exist — and you get AttributeError.

✅ Bonus: Clean Up and Reinstall Dependencies

If you've been switching versions and running into trouble, consider doing a clean reinstall:

bash
Copy
Edit

```
pip install -r requirements.txt  # inside the yolov5 directory
```

Ensure you still have:

bash
Copy
Edit

```
torch==1.8.0
onnx==1.9.0
onnxruntime
```

Let me know once you've tried v6.0 or v6.2 and I can help you test the ONNX model or prepare C++ post-processing code for OpenCV.

You said:

the conversion is done. i want to test the model with c++, visual studio 2015 and opencv4.5.0. how to install them and give me the source code.

ChatGPT said:

Updated saved memory

Great! Since your YOLOv5 model has been successfully converted to ONNX, here's how you can:

✅ 1. Set Up OpenCV 4.5.0 with Visual Studio 2015

📦 Step-by-step Installation:

✅ A. Download OpenCV 4.5.0 (Windows)

Go to the official OpenCV Releases

Download: opencv-4.5.0-vc14_vc15.exe (choose VC14 for Visual Studio 2015)

✅ B. Install OpenCV

Run the .exe — it will extract to something like:

C:\opencv\opencv-4.5.0\

✅ C. Configure Visual Studio 2015

Include Directories:

C:\opencv\opencv-4.5.0\build\include

Library Directories:

C:\opencv\opencv-4.5.0\build\x64\vc14\lib

Linker Input:

Add these .lib files:

vbnet

Copy

Edit

opencv_world450.lib

Copy DLLs:

Copy opencv_world450.dll from:

makefile

Copy

Edit

C:\opencv\opencv-4.5.0\build\x64\vc14\bin

Into your .exe build folder or C:\Windows\System32

✅ 2. C++ Code to Load YOLOv5 ONNX with OpenCV

Here's a complete working example (no GPU, CPU only):

📄 main.cpp:
cpp
Copy
Edit

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/dnn.hpp>
#include <iostream>

using namespace cv;
using namespace dnn;
using namespace std;

// Parameters
float confThreshold = 0.4;
float nmsThreshold = 0.5;
int inpWidth = 640;
int inpHeight = 640;

vector<string> getOutputsNames(const Net& net) {
    static vector<string> names;
    if (names.empty()) {
        names = net.getUnconnectedOutLayersNames();
    }
    return names;
}

void drawPred(int classId, float conf, int left, int top, int right, int bottom, Mat& frame) {
    rectangle(frame, Point(left, top), Point(right, bottom), Scalar(0, 255, 0), 2);
    string label = format("%.2f", conf);
    label = to_string(classId) + ":" + label;
    putText(frame, label, Point(left, top - 5), FONT_HERSHEY_SIMPLEX, 0.5, Scalar(255, 255,
255));
}

int main() {
    // Load ONNX model
    Net net = readNetFromONNX("yolov5s.onnx");
    net.setPreferableBackend(DNN_BACKEND_OPENCV);
    net.setPreferableTarget(DNN_TARGET_CPU);

    // Load image
    Mat frame = imread("test.jpg");
    Mat blob;
    blobFromImage(frame, blob, 1/255.0, Size(inpWidth, inpHeight), Scalar(), true, false);
    net.setInput(blob);

    // Inference
    vector<Mat> outs;
    net.forward(outs, getOutputsNames(net));

    // Postprocessing
    vector<int> classIds;
    vector<float> confidences;
    vector<Rect> boxes;

    float* data = (float*)outs[0].data;
    const int dimensions = outs[0].size[2];
    const int rows = outs[0].size[1];

    for (int i = 0; i < rows; ++i) {
        float confidence = data[4];
        if (confidence >= confThreshold) {
            float* classes_scores = data + 5;
            Mat scores(1, dimensions - 5, CV_32FC1, classes_scores);
            Point classIdPoint;
            double max_class_score;
            minMaxLoc(scores, 0, &max_class_score, 0, &classIdPoint);
            if (max_class_score > confThreshold) {
                int centerX = (int)(data[0] * frame.cols);
                int centerY = (int)(data[1] * frame.rows);
                int width   = (int)(data[2] * frame.cols);
                int height  = (int)(data[3] * frame.rows);
                int left    = centerX - width / 2;
                int top     = centerY - height / 2;

                classIds.push_back(classIdPoint.x);
                confidences.push_back((float)max_class_score);
                boxes.push_back(Rect(left, top, width, height));
            }
        }
        data += dimensions;
    }

    vector<int> indices;
    NMSBoxes(boxes, confidences, confThreshold, nmsThreshold, indices);
```

```
    for (int i : indices)
        drawPred(classIds[i], confidences[i], boxes[i].x, boxes[i].y,
                 boxes[i].x + boxes[i].width, boxes[i].y + boxes[i].height, frame);

    imwrite("result.jpg", frame);
    imshow("Output", frame);
    waitKey(0);
    return 0;
}
```

✅ 3. Build Settings Summary in Visual Studio 2015

Setting	Value
C++ Include Directories	C:\opencv\opencv-4.5.0\build\include
Library Directories	C:\opencv\opencv-4.5.0\build\x64\vc14\lib
Linker → Input	opencv_world450.lib
Executable DLLs	Copy opencv_world450.dll next to your .exe

🔄 Need Help?

If you’d like a sample .vcxproj, or help adapting the code to your model's custom classes, just say the word.