ssh -p 10113 junge82@nipg11.inf.elte.hu

passwd

sftp://nipg11.inf.elte.hu:10113/

ssh nipg11

Autoencoder 1000 inputs to image (zip)

Gan Hálók aliasing

Compresses sensing

Sw? Ganra

Net\_autoencoder

{

"name": "Regression inference",

"type": "python",

"request": "launch",

"program": "/home/junge82/NiftyNet/niftynet/application/regression\_application.py",

"args": [

"inference","-c",

"~/niftynet/extensions/autocontext\_mr\_ct/net\_autocontext.ini",

"--starting\_iter 0 --max\_iter 500"

],

"console": "integratedTerminal",

"stopOnEntry": true

},

python3 net\_autoencoder.py inference -c ~/niftynet/extensions/autoencoder\_demo/vae\_config.ini --inference\_type encode --save\_seg\_dir ~/niftynet/output/vae\_demo\_features --inference\_iter -1

net\_download dense\_vnet\_abdominal\_ct\_model\_zoo  
net\_segment inference -c ~/niftynet/extensions/dense\_vnet\_abdominal\_ct/config.ini

net\_download mr\_ct\_regression\_model\_zoo

net\_regress -c ~/niftynet/mr\_ct\_regression\_model\_zoo/config.ini

python3 net\_regress.py train -c ~/niftynet/extensions/mr\_ct\_regression/net\_isampler.ini

python3 net\_regress.py train -c ~/niftynet/extensions/autocontext\_mr\_ct/net\_autocontext.ini --starting\_iter 0 --max\_iter 500

Tensorboard logging

OpenCV - Open Source Computer Vision Library

Simple ITK - National Library of Medicine **Insight Segmentation and Registration Toolkit (ITK)**.

NiBabel -- NiBabel Access a cacophony of neuro-imaging file formats. Read / write access to some common neuroimaging file formats

skimage .io scikit-image -- *scikit-image* is a collection of algorithms for image processing

tensorflow==1.7 is supported by NiftyNet

**YAML** (*YAML Ain't Markup Language*) is a [human-readable](https://en.wikipedia.org/wiki/Human-readable) [data serialization language](https://en.wikipedia.org/wiki/Data_serialization_language)

Yes, the net\_regress.py application with the default\_monomodal\_regression.ini configuration file provided under /config will do 3D MR->CT semantic regression by sampling patches of size [64,64,64]

/home/gergely/NiftyNet/config/default\_monomodal\_regression.ini

2018-08-23 17:14:14.912490: I tensorflow/core/platform/cpu\_feature\_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA

## **What is this warning about?**

Modern CPUs provide a lot of low-level instructions, besides the usual arithmetic and logic, known as extensions, e.g. SSE2, SSE4, AVX, etc. From the [Wikipedia](https://en.wikipedia.org/wiki/Advanced_Vector_Extensions):

**Advanced Vector Extensions** (**AVX**) are extensions to the x86 instruction set architecture for microprocessors from Intel and AMD proposed by Intel in March 2008 and first supported by Intel with the Sandy Bridge processor shipping in Q1 2011 and later on by AMD with the Bulldozer processor shipping in Q3 2011. AVX provides new features, new instructions and a new coding scheme.

In particular, AVX introduces [fused multiply-accumulate](https://en.wikipedia.org/wiki/Multiply%E2%80%93accumulate_operation#Fused_multiply.E2.80.93add) (FMA) operations, which speed up linear algebra computation, namely dot-product, matrix multiply, convolution, etc. Almost every machine-learning training involves a great deal of these operations, hence will be faster on a CPU that supports AVX and FMA (up to 300%). The warning states that your CPU does support AVX (hooray!).

I'd like to stress here: it's all about **CPU only**.

## **Why isn't it used then?**

Because tensorflow default distribution is built [without CPU extensions](https://github.com/tensorflow/tensorflow/issues/7778), such as SSE4.1, SSE4.2, AVX, AVX2, FMA, etc. The default builds (ones from pip install tensorflow) are intended to be compatible with as many CPUs as possible. Another argument is that even with these extensions CPU is a lot slower than a GPU, and it's expected for medium- and large-scale machine-learning training to be performed on a GPU.

## **What should you do?**

**If you have a GPU**, you shouldn't care about AVX support, because most expensive ops will be dispatched on a GPU device (unless explicitly set not to). In this case, you can simply ignore this warning by

# Just disables the warning, doesn't enable AVX/FMA  
import os  
os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

... or by setting export TF\_CPP\_MIN\_LOG\_LEVEL=2 if you're on Unix. Tensorflow is working fine anyway, but you won't see these annoying warnings.

**If you don't have a GPU** and want to utilize CPU as much as possible, **you should build tensorflow from the source optimized for *your* CPU** with AVX, AVX2, and FMA enabled if your CPU supports them. It's been discussed in [this question](https://stackoverflow.com/q/41293077/712995) and also [this GitHub issue](https://github.com/tensorflow/tensorflow/issues/8037). Tensorflow uses an ad-hoc build system called [bazel](https://bazel.build/) and building it is not that trivial, but is certainly doable. After this, not only will the warning disappear, tensorflow performance should also improve.