

FP Project – Style Transfer

What is going on?

This is a basic implementation of this paper: “A Neural Algorithm of Artistic Style” by Leon A. Gatys, Alexander S. Ecker, Matthias Bethge (<https://arxiv.org/abs/1508.06576>).

Basically the paper demonstrates a way to combine an photo with a artistic style from another image. This works by creating a new image that minimizes the difference in the convolutional neural activations of the original image and the new image, and the difference in the convolutional neural activations of the style image and the new image. The new image combines aspects of both images.

For an explanation of how convolutional neural networks work refer to this video (<https://www.youtube.com/watch?v=JiN9p5vWHDY>)

For a thorough explanation refer to the paper, or the resources listed below (especially the video from Jeremy Howard).

Source Code Overview:

The source code is split into the following modules:

Lib: contains the main function that is executed when the program is run

JsonHandling: implements loading of .json files

ImageHandling: implements image loading, saving and resizing helper functions

Vgg16: contains definition of the VGG16 neural net (see resources)

Impl: implements the algorithm in tensorflow (the session function is what actually is executed)

Resources we used:

- Amazing explanation and implementation of the algorithm in python from Jeremy Howard (code: <https://github.com/fastai/courses/blob/master/deeplearning2/neural-style.ipynb>) (youtube: <https://www.youtube.com/watch?v=cRjPVN3oo4s> starting in minute 45)

- Introduction to Tensorflow in Haskell from <https://mmhaskell.com/tensorflow>

- Haddock documentation of tensorflow-haskell <https://tensorflow.github.io/haskell/haddock/>

- Haskell wiki

- VGG 16 implementation and weights in python using tensorflow, our vgg16 implementation is based on this one, we use their weights

Libraries:

- tensorflow-haskell library to use tensorflow in haskell (<https://github.com/tensorflow/haskell>)
- codec.picture : a haskell image library
- aeson : a haskell json library

Remarks

Tensorflow and haskell don't go well together. Understanding the computation graphs of tensorflow is hard enough in python, together with haskell's lazy evaluation it becomes much harder. Also tensorflow breaks the referential transparency, by allowing.

The tensorflow-haskell bindings are not in a ready state. Many functions of tensorflow are not implemented, and some actions that clearly break the graph don't not raise an error (adding a graphdef twice for example)

What will follow this project?

We will prepare multiple pull requests for tensorflow haskell, in order to help other people trying this library in the future.

Also we might clean the implementation and send it upstream, because one struggle we faced was, that there are virtually no usage examples of this tensorflow-haskell (just two very basic ones). One of us really wants to implement LBFGS or conjugate gradient optimisation routines in tensorflow haskell in order to get closer to the original results \