Digital Analysis of Paintings using TensorFlow

By Lasse Falch Sortland (Laf15)

Aim of the project
The aim of this project is to develop a Convolutional Neural Network (CNN). It will be able to take an image of a painting and classify as either being of a specific painter or not. Currently I am using a dataset of Sir Kyffin Williams (KW) paintings, but more may be added as time goes by. The process the images go through is 3 steps. The reason why it's called a convolutional neural network is that the first part of the process takes an image as a array of numbers and sends a filter over the image which sometimes are called a neuron or a kernel. The filter then moves along the image changing its weights based on the information it gets. The filter is also an array of numbers and the numbers are what we usually call weights. When we move the filter, or neuron, around the image that's called convolving. This is where the Convolutional Neural Network comes from as it is a network of neurons convolving around an image.

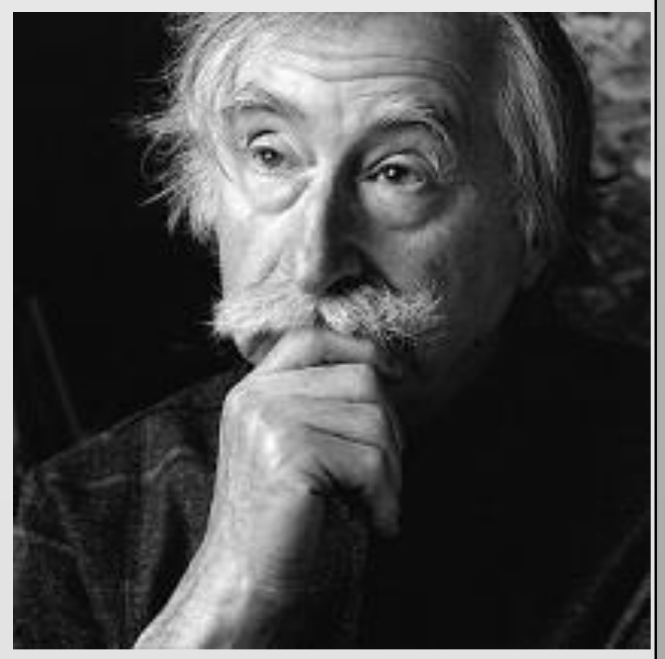

Figure 1 [1]: **Sir Kyffin Williams.** This is the artist behind the art that my convolutional neural network is currently being trained on.

As time goes by the aim is to add a Neural Style Transfer (NST) to get the ability to input images that are not of the KW style and add that style to it. NST uses the information that the neural networks can take out of a Neural Network (NN), usually a CNN and transfers that to another image. This takes the correlations of feature responses from the higher layers in the CNN to extract the style as a texture. So you take pairs of feature maps and you multiply them point wise and that gives you a correlation value which you can use to see how correlated those two features are to that specific image. When you do that with all the feature maps in the layer you get a correlation matrix. If you do that too all the layers in your model you end up with a texture model which can be used to model the style.

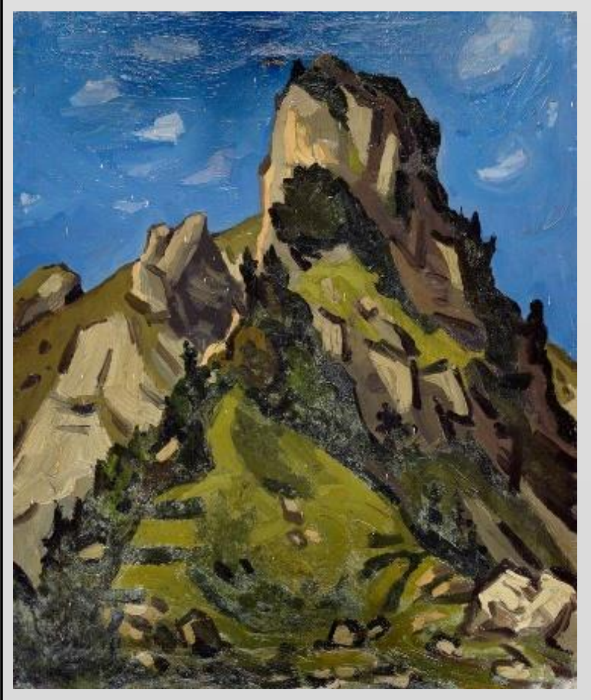

Figure 2 [2] [3]: **Alpine Mountain Landscape by Kyffin Williams**. This is one of the images in my dataset. As you can see the style of the image is quite idiosyncratic as he uses a palette knife rather than a brush. This makes his strokes quite distinct on the canvas. Making for a very recognizable style.

Most of the background work has been reading up on how to use TensorFlow(TF) and trying to figure out a way to make it work using the MNIST dataset. One of the biggest problems with this project so far has been the documentation of TF. There are a severe lack in proper, well structured documentation which makes it extremely hard to figure out how to fix something when you do something wrong. On top of that the error messages are very often quite ambiguous and makes for even harder to get a working piece of software. This has been a huge time sink for my project and way too much time has been spent trying to understand parts of TensorFlow that are simple but not documented anywhere.

Progress
The current working version is used to map integers from handwritten digits in the MNIST dataset. I am currently working on spoofing the dataset object that TF uses for the MNSIT dataset, as most tutorials online use the MNIST dataset object which is a different dataset object than the one you get if you read in the files how you are supposed to do. I have completed a set of scripts that turn a batch of images of any size into a specific size, and a script that turn those images from a folder of .png files to a single .tfrecord file.

TFRecords are the way that TF generally reads image files and is a compressed type of file that is specifically structured for TF to read. The problem with this file is that there is no read documentation on how it is supposed to be structured and how to read it into a TF NN which makes it very hard to use. This is why for this project, the decision has been made to spoof the MNIST dataset object which there are plenty of documentation on how to use.

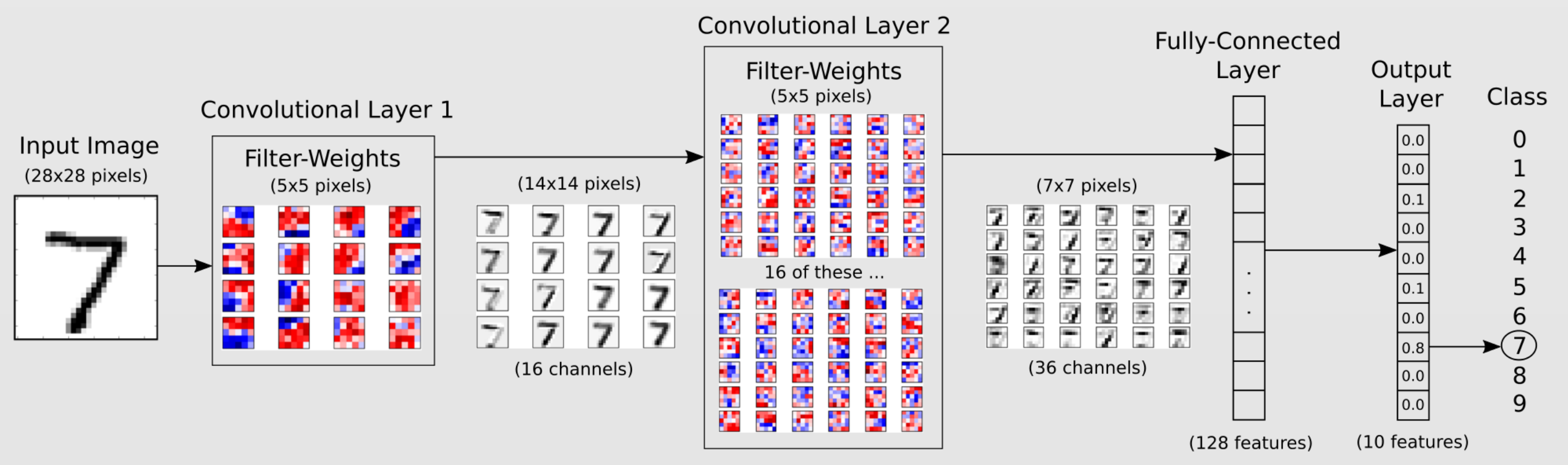

Figure 3 [4]: MNIST flowchart by Hvass-Labs. This is a handwritten 7 from the MNIST dataset, and here you can see the process of putting it through the CNN that Hvass-Labs Designed for Hand Writing Recognition. The input layer is an image of what you wish to detect, in this instance it's a hand written 7. The Convolutional Layer 1 is the filters that the Convolutional Network get out of that image after it's changed it's weights based on said image. The 14x14pxl image set is a representation of what those filters see. The Convolutional Layer 2 is the second layer of convolutions and represents the filters. The Fully Connected Layer has 128 neurons and that feeds into another fully connected layer called the output layer. The output layer has the certainty of a image being of a specific class in a array with as many classes as you have. For this it might be

0,	1,	2,	3,	4,	5,	6,	7,	8,	9
0.008,	0.171,	0.008,	0.003,	0.009,	0.017,	0.038,	0.728,	0.015,	0.003

So you can se there is some confusion between 1s and 7s however the CNN is 72.8% sure that it's a 7 in this instance.

Technical Information
The project is using the TF libraries as the library for doing NN related programming however it also uses OpenCV for image processing, sklearn for some machine learning functions, MatPlotLib for plotting and Numpy for large, multi dimentional arrays. The bulk of the project will be written in Python 3.5 using TF version 1.5.0. Currently the project contains the basis of the CNN written by Magnus Erik Hvas Pedersen [4] where I have removed the dataset and dataset object that he uses and designed my own. When that object is accepted by the TF functions I will start working on changing, rewriting and adding more functionality to the base CNN by Hvas. So far I have added a object that spoofs the one he's using in his CNN to make it accept my dataset rather than the MNIST dataset that he uses.

When the dataset object is completed it should be able to be used on the CNN that already are working for the MNIST dataset. Currently the data object should read the images in correctly however the labels are being a bit problematic as when the One Hot encoding is being done they get restructured into a format that TF doesn't like.

The object I have added has required me to figure out how the tensorflow.contrib.learn.python.learn.datasets.mnist.dataset object works and spoofing what it does to add my dataset to the CNN rather than the MNIST. The way the objects are supposed to be read into Tf is using a TFRecordReader on .tfrecord files which have a complicated structure that have next to no official documentation on how to use, how to create or how to understand. I have been in contact with someone else who does a similar project to mine, who have also been having the same problem with the .tfrecord files and I have decided to scrap tfrecords and spoof them instead.

There are very few tutorials on how to make a CNN working with a custom dataset, and generally the documentation for TF is rather lacking. One of the major problems is that in the official documentation a lot of the pages say for more information see here and they point to a different part of the documentation, however that part of the documentation then says the same thing as the first one and points back to that one, and you have no clue of what is what. Another problem is that unofficial documentation, tutorials and other information on how to use TF tends to be outdated or deprecated.

So far in this project I have spent about 30% of my time reading up on how to do something in tensorflow and another 30% of my time has been spent on trying to actually make that work realizing that one of the things has been changed or deprecated making the whole project wasted as there is no mention of what python version is used nor what TensorFlow version is used.

Further information
[1] Krizhevsky, Sutskever and Hinton [08/12/12]. ImageNet Classification with Deep Convolutional Neural Networks [Online]. Downloaded: [21/02/18]. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
[2] Anish Athalye [12/10/12]. neural-style [Online]. Downloaded: [08/02/18]. Available: https://github.com/anishathalye/neural-style
[3] Google LLC, Vincent Vanhoucke, Arpan Chakraborty [N/A]. Deep Learning [Online]. Downloaded: [29/01/08]. Available: https://eu.udacity.com/course/deep-learning--ud730
[4] No one author [N/A]. Kyffin Williams [Online]. Downloaded: [08/03/18]. Available: https://en.wikipedia.org/wiki/Kyffin_Williams
[5] A. D. Brown, G. L. Roderick , H. M. Dee and L. Hughes [2016] Visual digital humanities: National Library of Wales [Online]. Downloaded: [08/03/18]. Available: https://books.google.co.uk/books?hl=en&lr=&id=3AcUDgAAQBAJ&oi=fnd&pg=PA89&dq=%22Visual+digital+humanities%22&ots=BkBUnFhpg_&sig=Dx1661MO5PU24G3VlVhWB0fv5rM#v=onepage&q=%22Visual%20digital%20humanities%22&f=false
[6] A. D. Brown, G. L. Roderick , H. M. Dee and L. Hughes [09/01/14]. Can we date an artist's work from catalogue photographs? [Online]. Downloaded: [08/03/18]. Available: http://ieeexplore.ieee.org/document/6703803/?anchor=references
[7] Magnus Erik Hvas Pedersen [14/12/17]. Convolutional Neural Network [Online]. Downloaded: [08/03/18]. Available: https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/02_Convolutional_Neural_Network.ipynb
[8] Matt Lind [N/A]. Simple 3-Layer Neural Network for MNIST Handwriting Recognition [Online]. Downloaded: [08/03/18]. Available: https://mmlind.github.io/Simple_3-Layer_Neural_Network_for_MNIST_Handwriting_Recognition/
[9] The Codacus [N/A]. Prepare Real life Data Set To Train Your Tensorflow Model [Online]. Downloaded: [08/03/18]. Available: https://thecodacus.com/prepare-data-set-train-tensorflow-model/#.WqE4HOjFKUl

Work
The first thing I need to do is to make my dataset object work, currently it's accepting the images but not the labels but I have had some problems with understanding the error I've been given, and I have been trying to make you what they mean.
When I do get my dataset object working or find another way to add my dataset to the CNN I need to start working on optimizing the project, doing some tests with what gives the best results per minute and. Things I will be experimenting with are things like size, color scheme, orientations, etc.
When I have finished that part I will start working on NST and I have to start by reading up on how they work, to get a better understanding of them, try to see how other peoples implementations are structured, and then lastly implement one of my own, using the KW dataset.

Future Work
If I were to continue working on this project I would probably make it a more generic Neural Style Transfer program rather than one based on one artist. This would make it so that you could put in any input and style image and have the style taken from the first image and put it on the second. I think this technology is quite interesting and would like to do some more work with it.
I might also try to do some image processing prior to feeding it to the network to maybe try to highlight the style more or something more abstract like just see what happens if I take the two images and get the average of them or blur them or maybe transform them to look different and see how that would affect the output both the CNN and the NST

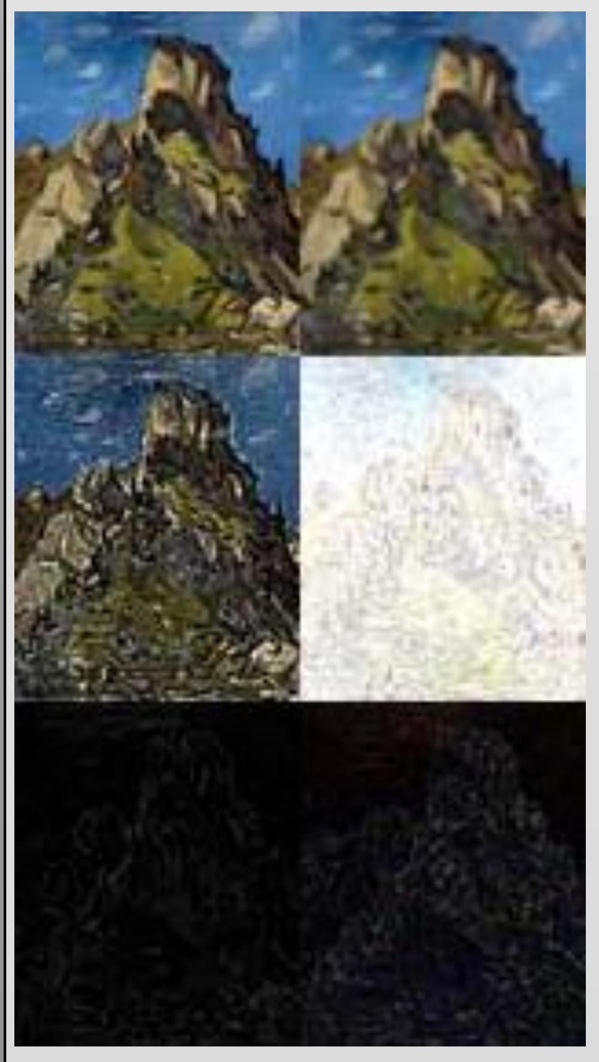

Figure 4[2] [3]: Examples of image processing methods I might use

From top right being 1, to top left being 2 middle right being 3:
1) Original
2) Original with gaussian blur
3) Original sharpened and darkened
4) Original sharpened and darkened and then divided by original
5) Original with gaussian blur subtracted by original
6) Original sharpened and darkened divided by original with gassian blurred subtracted from original

Here you can see some of the things you can do to images to get features out of them, four, five and six all are different edge maps and would make for a quite interesting style to add to an image if you imagine that the Black in five and six is treated as alpha and the white in four is treated as alpha.

For this I could do the image processing on the input images, do it on the filters or do it on output images, however I think the most interesting would be to do it to the filters directly followed by the input images. Doing this to the filters could cause the filters to become widely inaccurate and defeat the whole purpose of them or it could add more accuracy or a more distinct image.

For NST I would imagine it would add quite a bit of noise to the output image, however there is no way to know for sure without trying.