AARHUS UNIVERSITY

Visual Analytics portfolio

Emil Trenckner Jessen

20th May 2021

Contents

1 Introduction

This document provides an overview of the portfolio in the Cultural Data Science course Visual Analytics. The document is divided up with sections for the individual assignments that contains subsections providing information on 1) The assignment description, 2) Methods and 3) Results and discussion.

1.1 GitHub repository

Access to the GitHub repository can be found via the link here:

https://github.com/emiltj/cds-visual-exam

The repository contains the individual assignments a long with READMEs. The READMEs specify cloning the repository, setup (virtual environment installation and data collection) as well as the execution of the individual assignments. The information is also provided in this document.

1.2 Getting started

For running my scripts I'd recommend following the below steps in your bash-terminal (notice the bash scripts are different, depending on your OS). This functions as a setup of the virtual environment, as well as an execution of a bash script that downloads all the data to the data folders respective to the assignments.

Cloning repository and creating virtual environment

Listing 1: bash terminal - MAC/LINUX/WORKER02

```
git clone https://github.com/emiltj/cds-visual-exam.git
cd cds-visual-exam
bash ./create_vis_venv.sh
```

Listing 2: bash terminal - WINDOWS

```
git clone https://github.com/emiltj/cds-visual-exam.git
cd cds-visual-exam
bash ./create_vis_venv_win.sh
```

Retrieving the data

The data is not contained within this repository, considering the sheer size of the data. Using the provided bash script data_download.sh that I have created, the data will be downloaded from a Google Drive folder and automatically placed within the respective assignment directories.

Listing 3: bash terminal

```
bash data_download.sh
```

After cloning the repo, creating the virutal environment and retrieving the data you should be ready to go. Move to the assignment folders and read the READMEs for further instructions.

2 Image search (assignment 2)

2.1 Assignment description

Using the Oxford-17 data set, compare RGB-histograms of a target image and the rest of the image corpus. Utilize the chi-square method to the calculations one-by-one. Furthermore, have the script save a data frame with two column names: "filename" and "distance".

- Make sure to round the number to 2 decimal places
- Also find the image with the shortest distance to target image and print it in the terminal.

2.2 Methods

Specifically for this assignment

For solving the project task I have made use of several functions from the cv2 library; the most important being calcHist(). It was used in this assignment for accessing information about the intensity distribution of all 3 color channels. The histograms of non-target images and the target images were then compared using the chi-square method. As a bonus, to allow for an easy inspection of the results, I included the argument "save" in the script - this gives the ability of saving both the target image and the image closest in distance, although it was not specifically required in the assignment description (see images in directory "out/", or in the section "results and discussion", below.

On a more general level (this applies to all assignments)

I have tried to as accessible and user-friendly as possible. This has been attempted by the use of:

- Smaller functions. These are intended to solve the sub-tasks of the assignment. This is meant to improve readability of the script, as well as simplifying the use of the script.
- Information prints. Information is printed to the terminal to allow the user to know what is being processed in the background.
- Argparsing. Arguments that let the user determine the behaviour and paths of the script.

2.3 Results and discussion

Finding the closest image

Figure 1: Target image (left) and closest image (right)

By looking at the images above; target image (left) and the image with the smallest histogram distance (right), we see some clear similarities in terms of color and color intensities. The placement of the high intensities of yellow are similar, as is the color intensities of ground around it. This was however, not a given. When reducing the complexity of an image to its RGB histogram, we lose important spatial information about the colors as this method does not take placement of the different color intensities into account. Images with high pixel intensities in the bottom left for a given color will - using this method - be as similar to an image with the same high pixel intensities in the top left for the same given color (given that all other pixels are the same for both images). If we want to find images that are similar in a way that

is more in line with the human interpretation of the word "similar", you might want to consider using CNN approaches instead.

Output .csv file (distances_to_image_..csv)

After running the function with the default arguments, a new output .csv file is created. It shows the distances for the images in the image corpus to the target image, and clearly shows filename.

2.4 Usage

Make sure to follow the instructions in the README.md located at the parent level of the repository, for the required installation of the virtual environment as well as the data download.

Subsequently, use the following code (when within the cds-visual-exam folder):

Listing 4: bash terminal

```
cd assignment_2
source ../cv101/bin/activate # If not already activated
python image_search.py
```

2.4.1 Optional arguments

- "-f" "–filepath", type = str, default = os.path.join("data", "*.jpg"), required = False, help= "str - path to image corpus")
- "-t" "–targetpath", type = str, default = os.path.join("data", "image_0002.jpg"), required = False, help = "str - path to target file from which to calculate distance to the other images")

3 Logistic Regression and Neural Network benchmark MNIST classification (assignment 4)

3.1 Assignment description

Create two command-line tools which can be used to perform a simple classification task on the MNIST data and print the output to the terminal. These scripts can then be used to provide easy-to-understand benchmark scores for evaluating these models.

- Include two scripts; one with neural networks and one with logistic regression
- BONUS
  - Both scripts output a classification report to the terminal and saves it as well
  - Allow the user to determine number and size of layers for the Neural Network
  - Allow the user to determine parameters for the Logistic Regression
  - Allow the user to add an unseen data of any dimensions, process it, and let the classifier classify the new image
  - Allow the user to save the trained models for future use

3.2 Methods

Specifically for this assignment

Prior to training, both scripts had the data min-max scaled to allow for faster processing and better convergence. Both the training and test data was scaled using the values of the training data, to avoid information to flow from the training set to the test set. These scripts allows for simple model benchmarking on the

MNIST dataset, which could prove useful if one wanted to test other models for classification of the same data set. Moreover, all the "[BONUS]" features were included. This means that the user is allowed to specify parameters such as number of epochs (-ep) and hidden layers (-H) for the Neural Network (NN), while the arguments for the Logistic Regression (LR) script lets the user specify C-values (-c) and penalty method (-p). The classification reports can be saved if specified using the argument –save, and the user can furthermore predict any new images (regardless of dimensions and colors) using the argument –individual.

On a more general level (this applies to all assignments)

I have tried to as accessible and user-friendly as possible. This has been attempted by the use of:

- Smaller functions. These are intended to solve the sub-tasks of the assignment. This is meant to improve readability of the script, as well as simplifying the use of the script.
- Information prints. Information is printed to the terminal to allow the user to know what is being processed in the background.
- Argparsing. Arguments that let the user determine the behaviour and paths of the script.

3.3 Results and discussion

Classification reports

When looking at the results, the classification benchmarks for the LR classifier and the NN classifier seem to have similar performance, with a macro average f1-scores of .91 for both models.

	0	1	2	3	4	5	6	7	8	9	accuracy	macro avg	weighted avg
precision	0.96	0.93	0.91	0.89	0.91	0.86	0.96	0.93	0.89	0.91	0.91	0.91	0.91
recall	0.97	0.97	0.91	0.88	0.95	0.85	0.95	0.94	0.83	0.89	0.91	0.91	0.919
f1-score	0.97	0.95	0.91	0.88	0.93	0.86	0.95	0.93	0.86	0.90	0.91	0.91	0.91
support	198.0	232.0	194.0	225.0	165.0	171.0	180.0	203.0	198.0	234.0	0.91	2000.0	2000.0

Table 1: Logistic regression classification report

	0	1	2	3	4	5	6	7	8	9	accuracy	macro avg	weighted avg
precision	0.94	0.96	0.91	0.91	0.88	0.90	0.94	0.91	0.89	0.90	0.91	0.91	0.91
recall	0.95	0.97	0.91	0.91	0.92	0.84	0.93	0.93	0.90	0.88	0.91	0.91	0.92
f1-score	0.95	0.96	0.91	0.91	0.90	0.87	0.94	0.92	0.89	0.89	0.91	0.91	0.91
support	198.0	232.0	194.0	225.0	165.0	171.0	180.0	203.0	198.0	234.0	0.91	2000.0	2000.0

Table 2: Neural network classification report

NN's tend to outperform LR classifiers when both data and training time is plentiful and given the right hidden layer structure and parameter settings. As NN classifiers often take longer to train, the default parameters (hidden layer structure and number of epochs) were set to result in low runtimes for faster processing. More training would likely have resulted in the NN outperforming the LR.

When looking at the performances for the individual numbers, it becomes evident that the classifer did not classify equally well for all numbers. 3, 5 and 8 seem to be harder to predict - this is likely due to the similarity that they may have with other numbers (i.e. pixels of 3 and 5 might have a lot of overlap). When looking at the classification of an individual test image of the number 4 the LR script was capable of classifying it as 4, wheras the NN incorrectly classified it as 5. Again, this is with the default parameters - longer training for the NN may very well have resulted in a better prediction here, also.

In conclusion, the classification reports serve as benchmarks that might be used when evaluating new models for classification purposes: *how well does another model compare in comparison to the relatively simple models used here?*

3.4 Usage

Make sure to follow the instructions in the README.md located at the parent level of the repository, for the required installation of the virtual environment as well as the data download.

Subsequently, use the following code (when within the cds-visual-exam folder):

Listing 5: bash terminal

```
cd assignment_4
source ../cv101/bin/activate # If not already activated
python lr-mnist.py
python nn-mnist.py
```

3.4.1 Optional arguments

lr_mnist.py arguments for commandline to consider

- "-o" "–outname", type = str, default = "classif_report_neural_networks.csv", required = False, help = "str - containing name of classification report")
- "-s" "–save", type = bool, default = True, required = False, help = "bool - specifying whether to save classification report")
- "-i" "–individual", type = str, default = os.path.join("data", "test.png"), required = False, help = "str - specifying a .png file which is to be classified using this logistic regression model.")
- "-p" "–penalty", type = str, default = "none", required = False, help = "str - specifying penalty for the classifier - possible values: n̈one, l̈1, l̈2, ëlasticnet")
- "-c" "–c", type = float, default = 1.0, required = False, help = "int - specifying c-parameter for the classifier. recommended values: 0.01, 0.1, 1.0, 10, 100, where lower values mean stronger regularization")

nn_mnist.py arguments for commandline to consider

- "-o" "–outname", type = str, default = "classif _report _neural _networks.csv", Default when not specifying name of outputfile required = False, help = "str - containing name of classification report")
- "-s" "–save", type = bool, default = True, Default when not specifying required = False, help = "bool - specifying whether to save classification report")
- "-i" "–individual", type = str, default = os.path.join("data", "test.png"), required = False, help = "str - specifying a .png file which is to be classified using this neural networks model.")
- "-H" "–hiddenlayers", type = list, default = [8, 16], required = False, help = "list - specifying the hidden layers, each element in the list corresponds to number of nodes in layer. index in list corresponds to hiddenlayer number. E.g. [8, 16]")
- "-e" "–epochs", type = int, default = 50, required = False, help = "int - specifying number of epochs for training the model. Default = 50")

4 CNN classification of impressionist paintings (assignment 5)

4.1 Assignment description

Build and train a deep neural networks classifier to classify artists of impressionist paintings. Can a machine-learning algorithm classify the artist of an impressionist painting? Use either the architecture ShallowNet or LeNet.

- You should save visualizations showing loss/accuracy of the model during training
- You should also a save the output from the classification report.
- For reshaping images, I suggest checking out cv.resize() with the cv2.INTER_AREA method

4.2 Methods

Specifically for this assignment

Using a compact looped structure, the paintings of the individual artists were loaded into working memory. As the CNN we use requires data in the same format, the loaded paintings were resized and converted into the right format. To improve the versatility of the script, the user is given the option of choosing between either LeNet or ShallowNet, as well as specifying resized dimensions of the images (-d), batch size of the script (-b), and also number of epochs for training (-e). Classification reports are saved to the directory out using the argument -b, a long with a plot showing the architecture and a plot of the training history (the relationship between training epochs and the loss/accuracy of the model).

On a more general level (this applies to all assignments)

I have tried to as accessible and user-friendly as possible. This has been attempted by the use of:

- Smaller functions. These are intended to solve the sub-tasks of the assignment. This is meant to improve readability of the script, as well as simplifying the use of the script.
- Information prints. Information is printed to the terminal to allow the user to know what is being processed in the background.
- Argparsing. Arguments that let the user determine the behaviour and paths of the script.

4.3 Results and discussion

Classification reports

	Cezanne	Degas	Gauguin	Hassam	Matisse	Monet	Pissarro	Renoir	Sargent	VanGogh	accuracy	macro avg	weighted avg
precision	0.21	0.44	0.42	0.23	0.45	0.32	0.36	0.43	0.49	0.38	0.34	0.37	0.37
recall	0.43	0.15	0.37	0.38	0.20	0.24	0.45	0.44	0.31	0.40	0.34	0.34	0.34
f1-score	0.29	0.22	0.39	0.29	0.27	0.27	0.40	0.43	0.38	0.39	0.34	0.33	0.33
support	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	0.34	990.0	990.0

Table 3: ShallowNet architecture classification report

	Cezanne	Degas	Gauguin	Hassam	Matisse	Monet	Pissarro	Renoir	Sargent	VanGogh	accuracy	macro avg	weighted avg
precision	0.26	0.29	0.39	0.38	0.32	0.30	0.24	0.34	0.41	0.37	0.31	0.33	0.33
recall	0.10	0.26	0.34	0.17	0.23	0.42	0.60	0.33	0.36	0.35	0.31	0.31	0.31
f1-score	0.14	0.27	0.36	0.23	0.27	0.35	0.34	0.33	0.38	0.36	0.31	0.30	0.30
support	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0	0.31	990.0	990.0

Table 4: LeNet architecture classification report

As can be seen in the tables (using default parameters), similar performance were found when utilizing the LeNet and the ShallowNet architecture, with slightly higher perfomance for the less complex architecture, ShallowNet. It achieved a macro average F1-score of .33, compared to the score of .30 , that LeNet achieved. Baseline accuracy for 10 classes is at 10%, which means an increase of roughly 20 percentage points (not

all artists had the same number of paintings) compared to a model that classifies randomly. Paintings from artists such as Monet seems to be easier to classify, compared to artists such as Cezanne. Why might this by? Well for Monet for example, it might be that the high performance is due to Monet being quite consistent in almost always depicting French landscapes with roughly the same style of brush strokes.

Training histories

Training history of CNN following the ShallowNet architecture

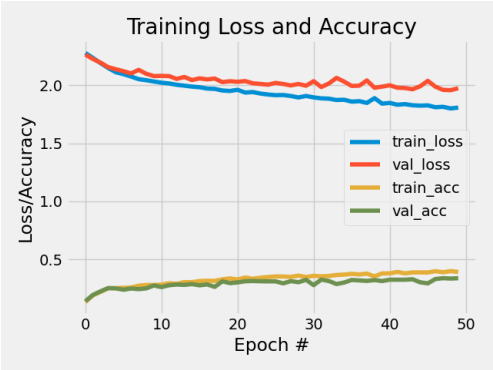

Figure 2: ShallowNet training history report

We see a steady climb in training loss and training accuracy - the more epochs the better performance on the training data. However, when training any machine learning classifier, we want the model to be able to generalize to new data. That makes the validation loss and validation accuracy more interesting. When looking at accuracy, we see that the validation accuracy starts to diverge from the training accuracy at around 5 epochs suggesting overfitting. Although additional training results in overfitting, our accuracy for the validation data set seems to be increasing slightly over epochs - even when the model starts to overfit. Although more than 5 epochs seems to generate an overfit model, they still provide for a better generalizable model. Using regularizations methods such as LASSO, Ridge regression or a combination (ElasticNet) could perhaps have reduced the overfitting, by shrinking weights of little importance during the training. A dropout layer might also have been utilized. However, if one wants fiddle with hyperparameter tuning to find the best performance, one should note that this might result in a model that overfits slightly to the validation data and not just the training data. Having a validation set and an additional test set, will enable hyperparameter tuning while still ensuring that the performance metrics are to be fully trusted.

Training history of CNN following the LeNet architecture

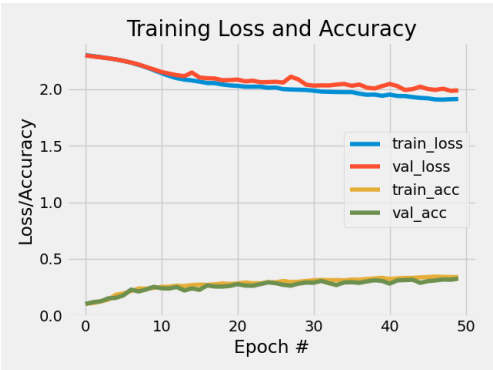

Figure 3: LeNet training history report

We see a different trend when using the LeNet architecture. Given the more complex architecture of the LeNet model, we have a model that does not result in much overfitting given the first 50 epochs. Given longer training, this model may very well have begun to achieve higher F1-scores than the other model. Given any future use this would be recommended a long with parameter tuning for optimal performance, and an additional test set to be able to accurately know the performance.

4.4 Usage

Make sure to follow the instructions in the README.md located at the parent level of the repository, for the required installation of the virtual environment as well as the data download.

Subsequently, use the following code (when within the cds-visual-exam folder):

Listing 6: bash terminal

```
cd assignment_5
source ../cv101/bin/activate # If not already activated
python cnn-artists.py --cnn "ShallowNet"
python cnn-artists.py --cnn "LeNet"
```

4.4.1 Optional arguments

- "-c" "–cnn", type = str, default = "ShallowNet", required = False, help = "str - specifying cnn architecture, use either "ShallowNet" or "LeNet"")
- "-r" "–resizedim", type = list, default = [32, 32], required = False, help = "list - specifying dimensions that the pictures should be resized to, e.g. [32, 32]")
- "-b" "–batchsize", type = int, default = 200, required = False, help = "int - specifying batch size")
- "-e" "–epochs", type = int, default = 50, required = False, help = "int - specifying number of epochs")

5 Stylized paintings and classification (self-assigned)

5.1 Assignment description

This self-assigned assignment has two main questions it seeks to investigate. The investigation is not hypothesis driven nor research oriented. Rather, it is meant to be a fun and atypical way of exploring some methods used in visual analytics.

Question 1 - Generating stylized paintings

Is it possible to use the method of style transfer to stylize paintings of one artist with the style of another? This assignment seeks to investigate the possibility of not just transferring style from a painting to an actual image as is so often done, but to explore the transferral of the style of one painting to another painting which already is painted in a particular style. Using paintings from the artists Cezanne and Monet from the impressionist paintings dataset, create a script which generates new stylized images. The stylized images should contain the contents of Cezanne with styles of Monet and vice versa.

- Save the stylized images in unique folders with names specifying the which artist the content and style came from.
- Save examples that clearly show the before and after result; save an image which includes content image, style image and the resulting stylized image.

Question 2 - Classification of stylized images

When CNN's classify paintings from artists, do they rely on the style of a given image? Or rather more on the content of the image? This part of the assignment seeks to train and test a classifier on the original paintings, and subsequently use the same trained model to classify the stylized paintings. Are Monet paintings with Cezanne style classified as Monet due to their content? Or rather classified as Cezanne due to their style? In other words, investigate the importance of content vs. style when classifying paintings.

- Use a pre-trained CNN classifier to distinguish between Cezanne paintings and Monet paintings.
- Use the same model to classify between the newly generated stylized images.
- Discuss the findings and consider whether the results tell us something general about either the classifier **or** about the stylization process

5.2 Methods

Generating stylized paintings

This script utilizes neural style transfer. Style transfer refers to the act of minimizing the distance between two sets of embedded images. One image (the style image) is embedded using the first few layers of a neural network. The other image (the content image) is embedded using the same neural network, but using the first many layers of the network. Likewise, the embedded image from this layer is also extracted. Using these two embeddings, style transfer then seeks to synthesize the two with regards to a loss function that minimizes the distance between the content embedding and the style embedding. Sometimes - as in the specific model use here - a Lagrange multiplier is introduced, which determines the weight of importance of the style embedding (level of stylization).

The original paintings were loaded and their order shuffled. They were then paired by their new shuffled index and each pair was then use to generate two new stylized paintings using the "magenta/arbitrary-image-stylization-v1-256" style transfer procedure. Content from painting A and style from painting B (and vice versa) were synthesized into stylized images. The stylized images were then preprocessed and a few examples of content + style + stylized images were saved to the examples directory in the directory "out". The entire script is designed to generalize to any other image corpora of any size. The script is furthermore designed so that you may specify both inpaths and outpaths.

Classifying paintings and stylized paintings

Both the original paintings and the stylized paintings were loaded using a self-define function and labels were automatically assigned. The paintings were then resized and formatted, as to match the input expectations of the subsequently defined pre-trained CNN model (MobileNetV2). The model was then trained to classify the artists of the original paintings, and later tested on an unseen subset of the same data. The same model was then set to classify the stylized images. The training history as well as the classification reports for both test sets were printed to the terminal and saved to the out directory.

On a more general level (this applies to all assignments)

I have tried to as accessible and user-friendly as possible. This has been attempted by the use of:

- Smaller functions. These are intended to solve the sub-tasks of the assignment. This is meant to improve readability of the script, as well as simplifying the use of the script.
- Information prints. Information is printed to the terminal to allow the user to know what is being processed in the background.
- Argparsing. Arguments that let the user determine the behaviour and paths of the script.

5.3 Results and discussion

Generating stylized paintings

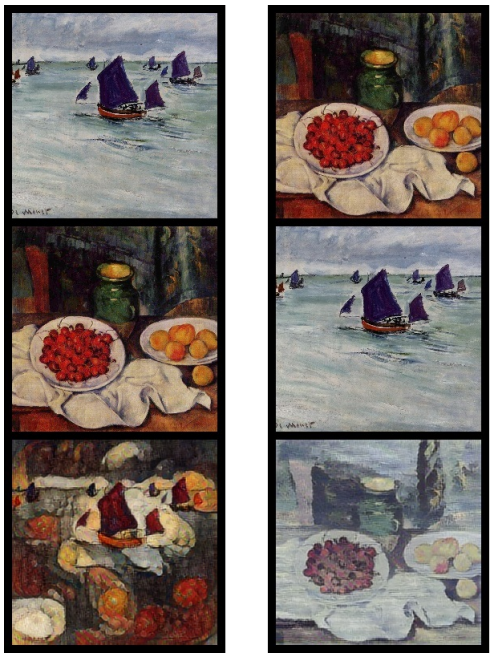

Figure 4: Example of the stylization process. Left: Monet content (top) and Cezanne style (middle) synthesization of the two (bottom). Right: Cezanne content (top) and Monet style (middle) synthesization of the two (bottom).

When looking at the above two images and (the rest of the examples), it does indeed seem possible to transfer style from a painting, to another painting. However, from a brief glance at the 20 examples it seems that when using portraits of people as style image, the process seems to generate paintings that are hard to interpret (e.g. image 12).

Alternatively to the random pairings of style/content images, one could have considered extracting the styles of all images of one artist and then subsequently found the weights resulting in the least information loss across all these images. This way we would have the general style of an artist to use as the style image when stylizing images. However, a caveat to this method would be the artist we have here, tend to not have the same style of painting over time. Moreover, the fact that the style embedding inevitably also extracts bits of content, would mean that one would have to acquire a very large number of paintings from an artist to model out the noise.

Classification of stylized images

From inspecting the classification report when predicting the stylized images, we can see that the classifier predicted roughly 25% of the Monet paintings with Cezanne style, as Monet (and vice versa).

	monet	cezanne	accuracy	macro avg	weighted avg
precision	0.92	0.97	0.94	0.95	0.95
recall	0.97	0.92	0.94	0.94	0.94
f1-score	0.95	0.94	0.94	0.94	0.94
support	101.0	98.0	0.94	199.0	199.0

Table 5: Original paintings classification report

	monet	cezanne	accuracy	macro avg	weighted avg
precision	0.21	0.28	0.25	0.24	0.24
recall	0.18	0.32	0.25	0.25	0.25
f1-score	0.19	0.3	0.25	0.24	0.24
support	497.0	497.0	0.25	994.0	994.0

Table 6: Stylized paintings classification report. **NOTE:** The Monet paintings with style from Cezanne had - for this classification - their True label set as "Monet".

Can we use this report for shedding light on the question of "*Are Monet paintings with Cezanne style classified as Monet due to their content? Or rather classified as Cezanne due to their style? Discuss the findings and consider whether the results tell us something general about either the classifier or about the stylization process.*"

There are two ways of interpreting these results. The first being that the model bases its predictions not so much on the content of the painting, but rather on the style. However, a confound comes in the way of going to this conclusion. Another way of interpreting the results takes the way in which the stylized paintings were generated into account. The classification results are likely the product of the fact that the stylized paintings that were synthesized were not 50% content and 50% style in the first place. Neural style transfer in its most basic form seeks to minimize the distance between the content of one image and the style of another - resulting in roughly half of each. However, the "magenta/arbitrary-image-stylization-v1-256" model uses a more sophisticated approach.

In summary, the classification report of the stylized paintings cannot lead to any direct conclusions. If instead a simple neural transfer model that stylized images with equal weighting to both content and style had been used, then we would have been able to make inferences about the relative importance that content and style had on the classifier.

5.4 Usage

Make sure to follow the instructions in the README.md located at the parent level of the repository, for the required installation of the virtual environment as well as the data download.

Subsequently, use the following code (when within the cds-visual-exam folder):

Listing 7: bash terminal

```
cd assignment_6
source ../cv101/bin/activate # If not already activated
python generate_stylized.py
python cnn-stylized.py
```

5.4.1 Optional arguments

- "-f" "–filepath", type = str, default = os.path.join("data", "*.jpg"), required = False, help= "str - path to image corpus")
- "-t" "–targetpath", type = str, default = os.path.join("data", "image_0002.jpg"), required = False, help = "str - path to target file from which to calculate distance to the other images")

6 Acknowledgements

I want to thank our instructors, Ross Deans Kristensen-McLachlan and Kristoffer Laigaard Nielbo for, in spite of a pandemic, managing to put together an awesome course, which was as well-structured as it was friendly.