
# CNN classification of impressionist paintings (assignment 5)

Emil Trenckner Jessen

20th May 2021

## Contents

# 1 Introduction

## 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 (including this assignment) a long with READMEs. The READMEs specify cloning the repository, setup (virtual environment installation and data collection). You may also read the information here.

## 1.2 Getting started

For running my script 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 folder and read the README for further instructions (or read a long here).

# 2 CNN classification of impressionist paintings (assignment 5)

## 2.1 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$_A$*REAmethod*

## 2.2 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.

| | 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 1: 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 2: LeNet architecture classification report

## 2.3 Results and discussion

### Classification reports

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

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

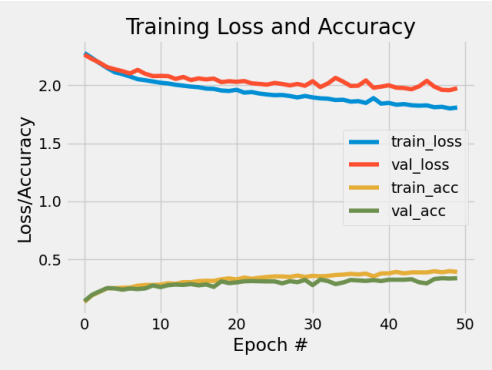


Figure 1: ShallowNet training history report

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**

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.

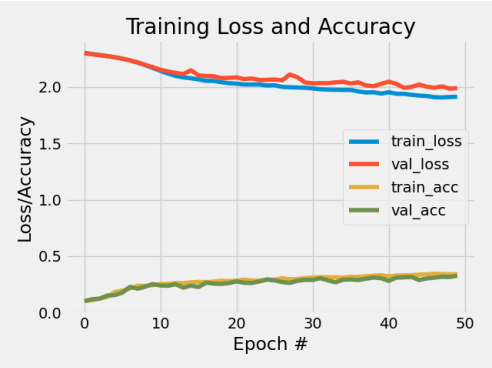


Figure 2: LeNet training history report

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.

## 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_5
source ../cv101/bin/activate # If not already activated
python cnn-artists.py --cnn "ShallowNet"
python cnn-artists.py --cnn "LeNet"
```

### 2.4.1 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")