ULiege: school engineering

Master thesis
Annotating artistic images with deep learning

Author:
Maxime Noirhomme Author
Pierre Geurts Supervisor
Matthia Sabatelli Supervisor

April 17, 2019

Abstract

TODO + where to put acknoledgments ?

Contents

Chapter 1

Introduction

In the past few years, deep learning framework becomes more and more powerful. This allows significant improvement in several fields as computer vision, speech recognition, natural language processing, machine translation and even more. One of the most known application is probably the imagenet challenge[3] which consists of 14 million of hand-annotated images to be well predicted. This allows a strong improvement in computer vision by the development of new deep neural networks, such as inception, vgg[10], resnet[7], deepnet and so on.

Nevertheless, this kind of architecture needs a lot of labeled training data in order to obtain great results. Moreover, get acquisition of labeled data is a very expensive task since this procedure can usually not be automatized (except in semi-supervised learning). Consequently, transfer learning has been used in order that other applications with smaller dataset benefit from these breakthroughs too. In particular, in computer vision, knowledge learns from imagenet have been transferred to other applications as .

Over the past decade, some museums have started to digitize their own artistic collection(s). Some examples of them are MoMA, the met collection, Colorado Digitization project, the Rijksmuseum Challenge and so on. Where the last one consists of 3 classification tasks on paintings, such as recognizing the author(s). The digitization allows better conservation, visibility, and understanding of artworks. This coupled with transfer learning have given great results. For example, in the paper Matthia Sabatelli et al.[9], they have shown that transfer learning from imagenet gives great results on the two first Rijksmuseum challenges for several deep convolutional neural networks (DCNN).

In this work, we will study the efficiency of transfer learning techniques on automatically find bounding boxes on objects on artistic images and classify them. However, since this task is difficult, the study will be focused on musical instruments. Indeed, the number of labelled data is small and the input space is complex due to the different styles. Contrary to natural images (photo), a specific instrument does not only have a specific shape or look. Furthermore, in this project, instead of directly learning a model on artistic images of instruments by using transfer learning from imagenet, an intermediate transfer learning is made before. A dataset from musical instrument museums online (MIMO)[1], which is a dataset that aggregates database of natural images of instruments from several museums. The used transfer learning techniques are detailed at .

Deep learning improvements saw the emergence of what we called style transfer methods. It consists to extract the style from one image and applies it to another one. In this document, we will propose approaches that use this framework in order to generate data augmentation. Where the basic idea consists at selecting one image from artistic images at random, extracts its style, applies it to other images (either artistic or natural images) selecting at random and repeats it for each batch. However, it's not possible for now in practice for computation reason (see blabla to understand why). Instead of this, a subset of the painting is selected and the data augmentation is pre-generated from it.

1.1 Problem definition

The problem we want to tackle in this paper can be divided into two sub-problems. First the definition of a classifier that is able to label artistic images with the right instrument name if it is represented as an instrument, otherwise label it as the negative class. Then, use this classifier in order to find bounding boxes of instruments in the whole image.

We consider the classification tasks where $\mathcal{X} = [0,1]^{W \times H \times 3}$ is the input space of rgb images ($W = width$ and $H = height$) and $Y = \{-1, 0, 1, ..., L-1\}$ is the set of $L+1$ possible labels (L is the number of different instrument considered). We also define the joined probability distribution of input and label space as $P_{\mathcal{X},\mathcal{Y}}(\mathcal{X}, \mathcal{Y})$ (for simplicity, $P(X, Y)$. Then the problem consists in training a classifier on a dataset $d = \{(x_i, y_i)\}_i^n \sim P(X, Y)$ (with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$) such that it minimizes the expected risk.

Chapter 2

Theoretical concepts

In this chapter, theoretical concepts as transfer learning and the proposed frameworks, the style transfer and its use for data augmentation, and the network visualization algorithm used in this work will be explained.

2.1 Transfer learning

2.1.1 Classical framework

2.1.2 adversarial transfer learning

2.2 Style transfer

Recently, a new framework has emerged from deep learning family, the style transfer. The purpose of this technology is to transfer the style of an image A to an image B by preserving the content of the image B. Two main techniques have been prominent, we will call them slow and fast style transfer(resp. [4] and [6]). The first one consists of an optimization problem using stochastic gradient descent that only involves the images A and B and a pre-trained feature extractor (basically VGG here) that is the same for all the styles. To do so, it started from a generated white noise image (C), and at each iteration, it tries to minimize (by modifying C), at the same time, both the style gap between A and C and the content gap between B and C. The content gap is modeled as the gap between the extracted feature of B and C. Where the style gap is modeled by building a style representation on the top of layers of interest ($\mathcal{L}$). The used style representation is define by Leon A. Gatys et al.[5]. The basic idea is to compute the correlation between filter response. It's done by computing the Gram matrix for each layer $l \in \mathcal{L}$ as follow:

$$G^l_{ij} = \sum_k F^l_{ik} \cdot F^l_{jk} \tag{2.1}$$

Where F^l_{ik} is the is the activation of the j^{th} filter at position k in layer l. Then, the style is optimized by minimizing the mean-squared distance between the entries of the gram matrices of the two images A and C. In order words, by minimizing (for layer l):

$$E_l = \frac{1}{4 \cdot N_l^2 \cdot M_l^2} \cdot \sum_{i,j} G^l_{A,ij} \cdot G^l_{C,ij} \tag{2.2}$$

Where N_l^2 and M_l^2 are respectively the number of filter and the filter size for the layer l.
And the total style loss is:

$$E_t = \sum_l w_l \cdot E_l \tag{2.3}$$

Where w_l is the associated weight of the layer l in the total style loss.
(TODO:fast style transfer)

2.2.1 Data augmentation

In our work, we studied the effect(s) of using style transfer images in order to increase our dataset. In a perfect world, the methodology would be to select two images at random, one for the style and the other for the content, and generates a third image. To make this interesting, this procedure needs to be called for each batch several's times. In other words, the style transfer algorithm needs to be called a lot of times during the training, which

is a problem for both slow and fast style transfer. The justification for slow one is quite straight forward, the order of magnitude is 1 generated image per minute for GTX 1080 Ti on 512 × 512 images (TODO: replace it with my computer + 224 x 224), which is clearly too slow. On the contrary, fast style transfer has an order of magnitude of about 40 ms per images for GTX 1060 Ti on 224 × 224 images, which increases the training time but in a reasonable way. However, if we consider that each image has a unique style, it will become unachievable. Indeed, since fast style transfer needs to train a network for each style, and a network take 4-6 hours for GTX 1080 Ti to train (TODO: replace it with my computer + 224 x 224), which is not acceptable.

Instead, we approximate this methodology by assuming that the input space is divided into n clusters, where each cluster contains images which share common characteristics about styles and building a style set $\mathcal{S} = \bigcup_{i=1}^{n} s_i$, where s_i is the common characteristics of the cluster i (called common style from now), i.e: the style of the most representative image of the cluster. Now, the idea is to select one image for the content and one s_i at random at each training iteration. Nevertheless, in practice, of course the entire input domain is not known, the determined cluster is an approximation.

In order to determine the clusters in an empirical way, we could proceed in a qualitative way by delimiting manually the cluster by examining these style. However, it is needed to be an art expert to perform this. Instead, we use the DBSCAN clustering algorithm that (Todo:small description of it). The metric that we used is the defined loss is the one defined at 2.3. The intuition behind this is that since style transferring achieve quite convincing results (at least in a qualitative looking) by using this loss, this loss should be meaningful. Consequently, if the loss between two images is low, then they probably have the same style s_i.

2.3 Visualization algorithm

Deep neural networks, and DCNN, in particular, are known to be black boxes. Which is a problem for two main reasons according to us. First, in the conception phase of the neural network, we want to know which part of the image the network uses to make its prediction in order to check if network do not use irrelevant information for its prediction (as background); or when the network is wrong, we would like to have more clue about how the prediction is made in order to debug the network. Then, in the user perspective, it's quite hard for an expert of a given field to accept an automatic answer without any justification or possible interpretation. That's one of the reasons why random forests are used in the medical field since they are more interpretable. In our case, we would like to give some elements that could help art experts to trust our results.

To do so, the visualization algorithm described in the work of Mariusz Bojarski[2] has been used. It is based on the fact that deep layers are the one with the most relevant information but with the lowest resolution. Thus, the idea is to combine them with shallow layers that have a bigger resolution but contains less useful information. To apply this idea, it first makes a forward pass on the network and saves the output of each Relu layer. After that, it takes the average of the deeper layer(L_n) and use a deconvolution operation on it in such a way that the result has the same shape than the average of the penultimate layer (L_{n-1}). And finally, the obtained scaled-up averaged feature map is point-wise multiplied by L_{n-1}, repeat it with the result and L_{n-2}, and so on until the end.

Chapter 3

Experimentation setup

In this chapter, the software, the dataset, the visualization method and the architecture used during this project will be detailed.[1]

3.1 Software

- Python 3.5.6
- keras 2.2.4
- tensorflow 1.10.0

3.2 Dataset

(TODO: I don't really like this paragraph) First, the imagenet dataset has been used indirectly since deep neural network pre-trained on imagenet have been involved in the beginning of all traning procedures during this work.

During this project, 4 main datasets have been directly used for training and testing architecture, the MIMO dataset, a dataset fetched from google, one from openImage challenge and an artistic dataset of musical instrument from Cytomine platform.

- The MIMO dataset, as described before, is about real photo of musical instruments. The dataset is composed of 52 828 images distributed among 75 classes. For the purpose of the this project, the study has been focused on a subset of 20 407 images distributed among 25 classes. In order to select them, classes that seem to be miss-classified or has too few sample with respect to their complexity have been removed. Then, among the leaving ones, the 25 classes have been taken in such a way that the resulting dataset contains, as much as possible, the most different type of instruments (string, rope, percussion instrument and so on). Nevertheless, there are 2-3 exceptions to that, for example saxhorn and tuba which are almost same classes, have been taken. This allows to analyze the abilities of models to distinguish between instruments that differs of only specific patterns.

  Furthermore, there are kind of color panels or other things to help the user to have information about the instrument(s). In order to be sure that models does not take them into account, the images have been cropped. An example of what the network used for making its prediction when it learned from vanilla and cropped data can be shown at .

- A dataset has been fetched from google by searching for each label in the MIMO dataset by searching "label_name" + " in painting". The search results to 150 images per class (i.e: $25 \cdot 150 = 3750$ images) but by removing bad images, the google dataset has 1614 images at the end of the process.

- The OpenImage dataset from OpenImage challenge has

- Cytomyne dataset

3.3 Style transfer images

3.4 Visualization tool

In order to visualize the attention map of trained neural network, github code from has been used. This allows to visualize model trained based on VGG19 architecture. During this work, this code has been slighty modified in order to work on both VGG19 and resnet50 architecture.

3.5

Bibliography

[1] Mimo dataset. http://www.mimo-international.com.

[2] Mariusz Bojarski, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Larry D. Jackel, Urs Muller, and Karol Zieba. Visualbackprop: visualizing cnns for autonomous driving. http://arxiv.org/abs/1611.05418, 2016.

[3] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database, 2009.

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. http://arxiv.org/abs/1508.06576, 2015.

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. http://arxiv.org/abs/1505.07376, 2015.

[6] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. http://arxiv.org/abs/1603.08155, 2016.

[7] Shaoqing Ren Kaiming He, Xiangyu Zhang and Jian Sun. Deep residual learning for image recognition. http://arxiv.org/abs/1512.03385.

[8] Jiirg Sander Xiaowei Xu Martin Ester, Hans-Peter Kriegel. A density-based algorithm for discovering clusters in large spatial databases with noise. https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf, 1996.

[9] Matthia Sabatelli, Mike Kestemont, Walter Daelemans, and Pierre Geurts. Deep transfer learning for art classification problems. https://www.clips.uantwerpen.be/ walter/papers/2018/skdg18.pdf, 2018.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. https://arxiv.org/abs/1409.1556.