Study on varied techniques for Artwork recognition

David Dinis
up201706766@edu.fe.up.pt
Henrique Freitas
up201707046@edu.fe.up.pt
José Gomes
up201707054@edu.fe.up.pt
Miguel Rosa
up201706956@edu.fe.up.pt

Faculty of Engineering
University of Porto
Porto, PT

1 Introduction

The combination of Computer Vision algorithms with Machine Learning strategies has largely boosted the range of tasks that a machine can accomplish. The application of Neural Networks to image-related problems was a big step in the way of technological progress but, as we will see, simpler methods - like the Bag of Visual Words - can also perform a relatively good job in these tasks.

This paper documents the process of creating a system that can perform an automatic classification of artwork shown in images. Two different classification problems will be explored - Multi-class and Multi-label Classification. For the Multi-class Classification problem two different approaches will be demonstrated: a simpler **Bag of Visual Words (BoVW)** approach and a relatively more recent solution with the application of a **Convolutional Neural Network (CNN)**.

There are already some robust implementations of CNN in image classification problems, like **VGG-16** [1], **GoogLeNet** [2] and **ResNet50** [3]. The CNNs implemented in this work have similarities with those, as they work pretty well and, besides being big architectures, have simpler blocks easy to understand and replicate.

2 Data Engineering

The dataset used is composed of images of artefacts that are part of the collection of The Metropolitan Museum of Art in New York, also known as The Met and was provided by the *iMet Collection 2019* competition. Each picture is associated with a single class for the Multi-class problem and one or more labels for the Multi-label problem.

2.1 Dataset class distribution

The analysis of the dataset shows a large imbalance in the class distribution. Some classes are represented in over 8000 images while the majority of them has a lot less examples, with cases of as few as 10 images. This is visually shown in Figure 1 below.

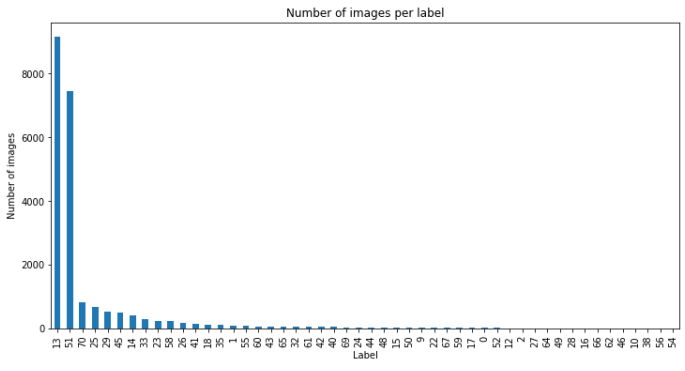

Figure 1: Amount of examples for each class in the dataset

2.2 Imbalanced dataset consequences

The problem with imbalanced datasets is in the way that the model perceives the environment and the distribution of the specimens. For us, humans, it is easy to differentiate objects accordingly to their visual features, like shapes, colors or textures, because we can logically cluster the similar objects and relate them in ways machines can't (at least at the time of writing).

As this work is partially based on the usage of **Neural Networks (NN)** and its goal is to define a general function to categorize all types of objects, having a lot of difference between the size of the sets will naturally lead the model to be biased and skewed towards the most frequent class examples. Although the **BoVW** doesn't directly suffer from imbalanced datasets, the collection of data is pre-processed before the train/test split, independently of whatever model it is used on, with the goal of minimizing the differences and having a better comparison between the approaches. In the 3.1 section, it is better explained how the dataset distribution can influence the performance of **BoVW**.

2.3 Balancing the imbalanced

Given the fact that we cannot successfully deal with an imbalanced dataset, the need to balance datasets arises. The techniques applied in this work to achieve this balance are **Over-sampling (OS)**, **Under-sampling (US)** [4] and a **3-class rearrangement** of the examples, as proposed by the professor.

2.3.1 Over-sampling - OS

This method balances the class distribution by duplicating the examples (as many times as necessary) of the least represented classes. Applying this bare technique in this particular dataset, the examples of the least represented classes (10 examples as seen in 2.1) would be individually repeated on average 800 times, to reach the 8000 examples of the most frequent classes. This repetition could easily induce the models to overfit in this over-sampled classes, thanks to the 800 repeated images.

2.3.2 Under-sampling - US

Opposed to **OS**, under-sampling balances the class distribution by randomly removing examples from the majority classes. This is also not an appropriate solution, as all classes would be left with only 10 examples. Without a reasonable amount of data to work with, the training might suffer from under-fitting, because the model can't make a good approximation of the pretended function.

2.3.3 3-Class Approach

During the development of this work, as an alternative, the professor proposed on rearranging the classes in only 3, as such: the 2 largely-represented classes stay the same, and the remaining are considered to belong to only one, resulting in only 3 different classes to differentiate. The goal when doing this conversion is to have a second classifier to differentiate between the remaining, joined-into-one classes. Ideally, if the first model predicts the image to be part of the third class, the second classifier would then predict which is the correct, original label (defined by

the initial dataset template). This makes it so that the third *virtual* class also needs to have a balanced representation of all the classes it abstracted.

2.4 The Chosen Methodology

With this setup, trying to balance (ironically) these techniques seemed like the proper way to prepare the data for the classifier's training. After some over-the-top experiments, 3 different balancing methods appeared to be right in the teams' perspective.

OSUS200 This is the first base approach that was decided to include in the comparison. This applies both OS and US, depending on the classes frequency, to have 200 examples of each class (the nomenclature is obvious now, hopefully). Classes with over 200 examples get under-sampled and classes under 200 examples get over-sampled.

OSUS500 Since 200 felt like a big compromise to the more than 8000 examples of the larger classes, an option with 500 examples per class was added, in order to have a better comparison.

OSUS3C Finally, for the last mix of techniques, this one actually uses all of them. One function was implemented with the purpose of easily transforming the original dataset into this template. The algorithm was created taking into account that the joined classes also need to be balanced between them. In this specific case, the 2 biggest classes ended up with 7000 examples and the remaining with 150 each (all the other 48 classes * 150 = 7200 examples).

Data details After the data preparation, the engineered dataset was split in train and test datasets, in a 70%/30% proportion. In a way to keep the consistency between the **BoVW** and **CNN**, when the images are loaded into memory, each one is resized to 300x300 pixels (CNN specifically imposes a fixed size input).

3 Multi-class Classification

3.1 Bag of Visual Words

The **Bag of Visual Words** approach closely resembles a common document classification solution called **Bag of Words (BoW)**. The **BoVW** consists on extracting key-points and descriptors from each image in the dataset. These key-points and descriptors represent features of each image. A clustering operation is then performed in order to group together similar features and create a vocabulary of visual words. With this vocabulary the signature of each image is created by collecting every word present in each image into an histogram that will be used to train a classifier - in the case of this paper a Support Vector Machine was utilized.

Trying to simplify this process, one can think, for example, of an image containing one isosceles triangle. The stage where it extracts the key-points and descriptors would likely detect the three acute angles present (consider these features as "top", "left" and "right"). It's then possible to represent this image by saying it contains one "top" feature, one "left" feature and one "right" feature. Inversely, in an image containing exactly these 3 features (not necessarily the same image as the original) we can probably guess that it is an isosceles triangle. The same way, if the program is presented with images of squares, it will likely detect four right angles, creating thus the "knowledge" that if an image contains 4 right angles it is an image of a square.

The example above is a very crude simplification of what happens (ignoring some important steps like clustering and training the classifier) but demonstrates the correct line of thought of the **BoVW** approach. Still, this approach, although simple enough to understand and implement, comes with various limitations. Some limitations include but are not limited to: heavy unnecessary calculation of features that are not relevant (background details that should not contribute for the classification for example); and because this approach stores features in an unordered manner, all spatial information is lost including spatial relationships between features (in this triangle simplification, an image containing three acute angles not connected to each other and in a random distribution over the image could still be recognized as an isosceles triangle - Figure 2).

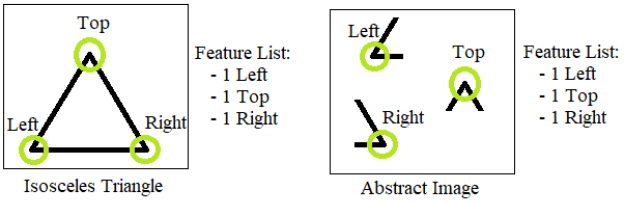

Figure 2: Example of an image that would hypothetically be classified as an isosceles triangle in our simplification scenario

Implementation details Having now a more concrete understanding of the **BoVW** in its theory, the following sections proceed to a more detailed explanation of what actually is done. The next section will be dedicated to explaining the what, how and why of the project.

Starting by the chosen feature detector, this project implemented the **KAZE** detector. The **Kaze** detector is a very accurate but also comparatively slow (in relation to the **ORB** detector for example). Since this project was run in an external server, processing power and execution speed were not the priority so the **KAZE** detector accuracy weighted a lot more in the decision process.

The next step was to cluster the list of descriptors detected by the **KAZE** detector into a vocabulary. For this, the **OpenCV BOW KMeans Trainer** was utilized because of its ease of use and also because it was previously used during the course. As for the number of clusters to create, the number 100 was arbitrarily chosen, meaning that the **OpenCV BOW KMeans Trainer** will create 100 different clusters from all the image descriptors. In a more rigorous environment the number of clusters to chose should be more carefully chosen. The amount of clusters utilized can influence the classification task: too few and very distinct features will be grouped in the same cluster, too much and slight variations of a feature may be wrongly classified as something entirely different.

Then comes the conversion of the notion of "image" to the actual **Bag of Visual Words**, or in this case an histogram of words for each image. This task consists of, for each descriptor of each image, calculate the word that it corresponds to in the vocabulary that was retrieved from the clustering. This project used **Scipy's vq() method** that does a quick work in a task such as this.

Finally, the last task of the actual **BoVW** method is to train a classifier that performs the actual classification of images and to which the image word histograms passed. In this project a **Support Vector Machine (SVM)** was utilized. The choice to use this classification method was not based on any previous knowledge. The project could have used a **K-Nearest Neighbor** approach or a **Random Forest** (or any other existing and possibly better approaches) but due to time constraints and since the **SVM** worked very well without much effort the project stuck to this approach. In a more realistic scenario various classification methods could and should be tested and the chosen method should be the one that best fits the necessities of the project - considering various factors such as the type of data to be classified, the speed of training, how fast it classifies new data, how well is the data classified, etc. In this choice, the fact that the data is imbalanced could have been a more important factor: this is the phase where an imbalanced dataset can skew the classification model. With the dataset's class distribution initial problem put aside, as discussed in 2, this was no longer a problem, so the **SVM** was a safe choice.

Performance measurements In order to measure the performance of the **BoVW** the dataset separated for testing was utilized for a comparison analysis. Basically the same methodology as the training was performed (descriptor extraction and histogram of words representation of the images) but, instead of training another classifier, the histograms were passed to the existing **SVM** as a list of images to predict the class of. With

the predictions of what our classifier believes the images are, it is possible to compare them to the real classes the images belong to and assess how accurately the system performs. A complete analysis of the results can be found in section 5.1.

3.2 Convolutional Neural Network

In the CNN approach, the previously defined 300x300 pixels of size are a good balance between performance and quality. To create and train the neural network the **Keras** framework from the **TensorFlow** package was used because of the large community, simplicity and familiarity from previous usage in this course.

For the model, 4 **Conv2D** layers were used with increasing filters number and in between each one a **BatchNormalization** layer was applied followed by a **MaxPooling2D** and a **Dropout**. Batch normalization applies a transformation that maintains the average output close to 0 and the output standard deviation close to 1, for each batch. The MaxPooling2D downsamples the tensor shape and this helps to prevent over-fitting while reducing computational load. The spatial information of the features is also eliminated as the feature extraction phase doesn't have the possibility to retain information about the area where the relevant data is located. To further prevent over-fitting a Dropout layer was added. This kind of layer randomly drops some connections between layers, by zeroing those chosen input values.

After these 4 *composed blocks*, a **Flatten** layer makes the conversion between this convolutional-based stage and the following "classification" stage, executing, as the name indicates, a *flattening* of the input. This means that whatever the input shape is, this layer will output the data in a 1-d structure (simply put, in a list).

This data structure is then passed on the **Multilayer Perceptron (MLP)** stage, composed of 4 **Dense** layers, that are capable of much more complex functions (mapping input to output) and having a better fit of the data, improving the classification results.

As for the activation functions, all layers use a standard **Relu activation**, except for the last Dense layer, that uses a **Softmax activation**.

4 Multi-label Classification

For the multi-label classifier, an adaptation of the Convolution Neural Network applied in section 3.2 is used. The main difference between both models is the activation function and the labels of the images.

The main difference between a multi-class and a multi-label classification problems is in the number of classes per example. In the previous multi-class problem, each image is bound to only one class. On the other hand, a multi-label problem have examples with multiple labels. To solve this problem, a technique called **One Hot Encoding** transforms the (possibly multi-) labels provided by the dataset into a binary matrix, where the 1's represent the labels a specific image belongs, and 0's otherwise.

The last Dense layer is also changed to have a **Sigmoid activation** function because it calculates the probability of the image being part of each class individually - does not constrain the classes to be mutually exclusive. The Softmax activation function is well suited for mutually-exclusive classes (which is not the case in a multi-label problem).

Apart from what is explained above, the rest of the CNN classifier is indistinguishable from the one created in Section 3.2, layer by layer.

Regarding the dataset, problems emerged when trying to import the whole **Kaggle** dataset due to memory limits of Google Colab (platform used for the project development). Due to this restriction, a smaller dataset was considered, although this had little in common with the original multi-label dataset. This problem led the team to use a smaller than desired set of images, which did not result in a great fit for the classifier, as it is expected when using small datasets.

5 Results

Various metrics were taken in consideration in order to better measure the quality of the implemented systems. The metrics evaluated were:

- Accuracy - Number of correct predictions over total number of predictions
- Precision - Ratio of positive identifications that were actually correct
- Recall - Ratio of actual positives that were identified correctly
- F2 Score - Harmonic mean of precision and recall, in this case giving more importance to recall than to precision. This has the effect of prioritizing the minimization of false negatives (results that are actually positive but are wrongly classified as negatives)

5.1 Bag of Visual Words Results

Dataset	OSUS200	OSUS500	OSUS3C	US	NB
Accuracy	0.621	0.653	0.529	0.180	0.434
Recall	0.619	0.653	0.529	0.207	0.021
Precision	0.567	0.609	0.529	0.175	1.018
F2 Score	0.597	0.635	0.526	0.182	0.017

Table 1: Collection of results gathered from the **BoVW** method with different balancing methods. US = Under-sampling to 10 examples for each class; NB = No Balancing strategy applied

By analysing the results in Table 1, the overall conclusion that came to be was that as long as the dataset is somewhat balanced (like **OSUS200** & **OSUS500** & **OSUS3C** and unlike **NB**) and has enough data (contrary to **US**), the **BoVW** approach will yield a similarly performing classification. Another takeaway is that the **OSUS500** took a considerable amount of time longer to complete training than the **OSUS200** and its performance was not that much better so increasing the dataset size by a factor of 2.5 (albeit the *OSUS* approach duplicates many examples) did not seem valuable with this method.

5.2 Convolutional Neural Network Results

Dataset	OSUS200		OSUS500		OSUS3C	
Epochs	100	200	100	200	100	200
Accuracy	0.815	0.820	0.923	0.923	0.782	0.827
Recall	0.818	0.823	0.923	0.923	0.781	0.827
Precision	0.815	0.816	0.923	0.921	0.782	0.830
F2 Score	0.815	0.820	0.922	0.922	0.781	0.826

Table 2: Collection of results gathered from the **CNN** classifier with different balancing methods

With these results in mind it is apparent that the CNN performed really good given that the dataset is somewhat balanced. The dataset with the best results was the **OSUS500** but since the **OSUS200** has a considerable less amount of data the results are acceptable with a faster training process. The **OSUS3C** was the worst of all of them, even though it had the most amount of data to train per class and needed the most training, as found by the results from 100 and 200 epochs that are so different. It was noted that the **CNN** gave considerable better results than the **BoVM**.

5.3 Multi-label Classification Results

The results of the multi-label were what we were expecting considering the small dataset used on this CNN despite the fact that there is no data to compare this approach to.

Epochs	100	200
Accuracy	0.57073	0.56220
Recall	0.12899	0.12970
Precision	0.21852	0.18645
F2 Score	0.13517	0.13517

Table 3: Multi-label Classification results

5.4 Weaknesses in the system

Adversarial images are a possible weakness of the system. An adversarial image is an instance with small, (maybe) intentional feature perturbations that can cause a machine learning model to make a false prediction. A possible solution to this problem is to incorporate some adversarial examples in the training set.

6 Conclusion

This project demonstrates a few of the existing methods for solving Multi-class and Multi-label classification problems with images. The project was considered to be successful since all developed classifiers could function properly according to expectations. The goal of the project was not to create a system that competes with state-of-the-art technology but to demonstrate the concepts behind Machine Learning algorithms in Computer Vision methodologies.

6.1 Future improvements

This work is not at all perfect having a lot of possible improvements and experimentation. All the various other detectors and classifiers mentioned in Section 3.1 could be experimented with and their performance compared. The CNN's also have a lot of room for experimentation either with hyper-parameters of each layer or with the actual layers themselves. Regarding the Multi-label Classification problem the main improvement to be made is to utilize a more complete dataset with more examples and diversity. Further exploration of the *OSUS* methods also seems like an area that could be improved.

6.2 Authors notes

Before choosing between Multi-label classification and Art-work detection the team weighted pros and cons of each tasks. The Art-work detection task proved to be the bigger challenge, not because of its inherent complexity but more due to the lack of proper data. To conceive a new dataset that would yield enough results in the limited time available would consume all the teams resources with few guarantees that it would actually work. On the other hand, the available dataset for the multi-label classification and the hardware resources for running such task were not very inspiring either. In the end, the team decided to move forward with the Multi-label Classification problem as the actual performance of the system was not the main objective of the project but rather the methodologies followed and the basic concepts of Machine Learning and Computer Vision.

References

[1] Simonyan, Karen; Zisserman, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition
https://arxiv.org/abs/1409.1556

[2] Szegedy, Christian; Liu, Wei; et all: Going Deeper with Convolutions
https://arxiv.org/abs/1409.4842

[3] He, Kaiming; Zhang, Xiangyu; et all: Deep Residual Learning for Image Recognition
https://arxiv.org/abs/1512.03385

[4] Brownlee, Jason: Random Oversampling and Undersampling for Imbalanced Classification
https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

[5] Andersson, Oskar; Marquez, Steffany Reyna: A comparison of object detection algorithms using unmanipulated testing images
http://www.diva-portal.org/smash/get/diva2:927480/FULLTEXT01.pdf

[6] Sharma, Sagar: Activation Functions in Neural Networks
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6