Artwork Recognition

Eduardo Carreira Ribeiro
up201705421@edu.fe.up.pt
Henrique Santos
up201706898@edu.fe.up.pt
Mário Gil Mesquita
up201705723@edu.fe.up.pt
Miguel Delgado Pinto
up201706156@edu.fe.up.pt

Faculdade de Engenharia
Universidade do Porto
Porto, PT

Abstract

This article presents the implementation of a system that can perform classification of images, applied in this case to the collection of The Metropolitan Museum of Art in New York. Such system has many practical applications, as it allows for automatically detecting artwork belonging to that collection. The proposed solution is described in detail, covering all of the methods and techniques that were explored. The overall performance of the system and situations where it may have reduced performance are also discussed. Results of the evaluation using adequate metrics are presented and, finally, conclusions are drawn and future improvements are debated.

1 Introduction

Image classification is a fundamental task in Computer Vision that allows attributing a set of classes or labels to an image. In this project, such techniques are applied to develop a system that can perform an automatic classification of artwork belonging to the collection of The Metropolitan Museum of Art in New York. For this aim, 2 state of the art alternatives were explored: classification using a bag of visual words representation, generated with K-means clustering; and deep learning methods based on CNNs, such as AlexNet.

2 Proposed solutions

2.1 Multiclass classification

The first goal consisted in classifying the artwork images, by attributing a single class to each of them. For this end, the two following approaches were considered:

2.1.1 Approach #1: Bag of words And Classifier

A first approach consists in generating predictions by training a classifier model that receives as input some type of data that characterizes the images. For this approach, a Bag of Words (BOW) descriptor was computed and used as input for the classifier, for each image. The architecture itself consisted of experimenting with a classification model based on Support Vector Machines (class SVC [1] on SKLearn).

This process can be divided in several steps. First, extracting all of the keypoints and descriptors from each image, using a KAZE keypoint detector and descriptor extractor. The objective of this first step is to gather all of the necessary image data needed to generate the vocabulary of visual words.

The next step consisted in using the BOWKMeansTrainer class in OpenCV to perform a clustering operation on all of the image descriptors, using a K-Means approach, in order to generate the visual words that will make up the vocabulary. This vocabulary is characterized by the resulting cluster centers. For this operation, the only parameter that was needed to set was the desired number of clusters, which, in this context, is the number of intended visual words. In this experience, by experimentation and also due to computation-power constraints, the K-Means operation was used to generate 100 visual words.

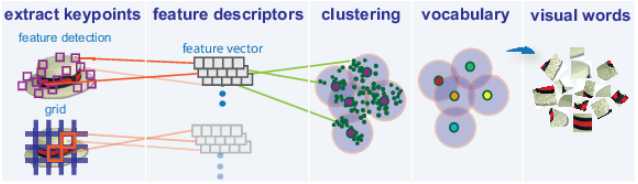

Figure 1: Image explaining the sequence of steps performed to generate the visual words, using the image dataset.

After building the visual vocabulary, the next step consists in generating a Bag of Words descriptor for each image. This is done by using several OpenCV classes and concepts, like a FlannBasedMatcher and a BOWImgDescriptorExtractor, that allows to extract all of the keypoints for a specific image, and use them to calculate the number of occurrences of each visual word in that image. Using that information, it's possible to generate a Bag of Words descriptor for the image.

These BOW descriptors are then directly used as input to the SVM classifier. From here on, several general machine learning techniques are used to ensure that the model's training and results are efficient and adequate. The dataset is split into training and testing sets, and a grid search cross-validation (GridSearchCV on SKLearn) is performed in order to find out the best metaparameters that work best for the model. The generated model is then trained using the training data, and used to generate predictions on the test set.

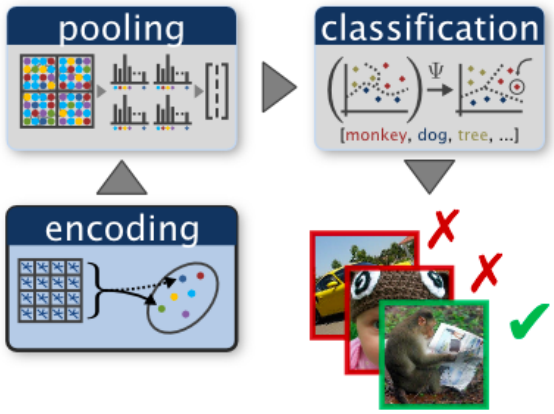

Figure 2: Image showcasing how the BOW descriptors for each image can be generated using the visual words, and used in a classification task to predict the label of the image.

To evaluate the performance of the model, the several evaluation techniques and metrics were calculated:

- Precision, Accuracy, Recall, F1-score, Support
- Macro-average and Micro-average
- Confusion Matrix

	Precision	Recall	f1-score	#
micro avg	0.51	0.51	0.51	4418
macro avg	0.02	0.03	0.02	4418
weighted avg	0.38	0.51	0.43	4418

Figure 3: BOW Descriptors + Classifier Testing Metrics

The first thing that one might notice when observing the testing results is the big difference between the micro-average and the macro-average results. That occurs due to the big class imbalance in the used dataset. Indeed, in the test set used for calculating the testing metrics, the number of images that had label 13 and 51 are above 1000, but for other classes the number of examples were near 100, and in some cases it was less than 10. Due to the fact that the model was trained with a big number of sample images with label 13 and 51, and with a small number of images with the remaining samples, the model makes better predictions for images with the former labels, and struggles to correctly predict the rest. Because the macro-average measure calculates the average of the metrics for each class, the final results are low. The micro-average, however, sums up the individual true and false positives and then calculates the metrics; this strategy is preferable if there is a class imbalance problem.

Overall, the performance of the model could be better, but the results are satisfactory having into account the fact that more complex techniques can be applied, like convolutional neural networks, that are at the same time more efficient, encompassing the feature extraction process in the training process, by using deep learning techniques.

2.1.2 Approach #2: Convolutional Neural Network

In this approach, a Convolutional Neural Network was used to classify the images with a single label. The chosen architecture for the network was very similar to an AlexNet [2], only diverging in the use of Dropout layers in order to prevent over-fitting to the training set. Initially, the group considered other, more modern, architectures (such as a VGG-16 [4]), but quickly found that training these deeper, narrower networks took its toll, in both time and memory. On the other hand, when using a shorter, wider network, it was possible to produce acceptable results. The last layer consisted in a dense layer with 3 and 48 nodes respectively for each CNN implementation, with a softmax activation. This makes the output a series of probabilities that accumulate to 1, and the maximum probability was taken for the class choice. The loss function used was a categorical cross-entropy metric.

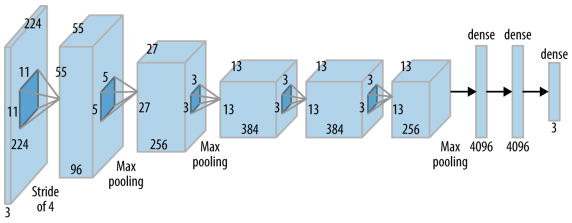

Figure 4: Used architecture for initial 3-class CNN.

The hyper-parameters used for the network were as follows:

- BATCH_SIZE = 32 (lower batch size due to memory constraints)
- EPOCHS = 20 (the network seemed to converge at this point)
- LEARNING_RATE = 0.0001 (low learning rate, slower convergence but less variance)
- HEIGHT = 224
- WIDTH = 224
- CANAL = 3

Initially, the network seemed to plateau at around 40% accuracy. When analyzing the confusion matrix, almost every image that was not of classes 13 or 51 was being classified as one of them. After analyzing the dataset, the group noticed that there was an enormous class imbalance between the classes aforementioned and the remaining ones. From the training dataset of 19999 images, 8263 were of class 13, 6745 were of class 51, and the remaining 4991 images were scattered among the remaining 48 classes. This caused the network to only learn features from these two dominant classes and always predict one of them, which is obviously not ideal from a generalist point of view. From the following confusion matrix, it's possible to observe exactly that: out of 2097 guesses, only 101 were classified as a different class (and only 30 were correct!), when, in reality, there were 717 images from a different class in the testing set.

$$\begin{array}{r} \textit{Class } 51 \\ \textit{Class } 13 \\ \textit{Others} \end{array} \begin{bmatrix} 361 & 110 & 26 \\ 192 & 651 & 45 \\ 301 & 386 & 30 \end{bmatrix}$$

Figure 5: Initial attempt's confusion matrix

Three possible approaches were devised to tackle this class imbalance problem:

- **Data Augmentation**[3]: By creating new images from the images at hand, using simple transformations such as lateral/vertical translations, rotations or zooming in, it was possible to expand the dataset with more examples of lesser represented classes, making the dataset more diverse;

Figure 6: Example of data augmentation on image (original: top left).

- **Class Weighting**: By calculating class weights before training, it's possible to value the few examples of less represented classes more than the ones with many examples of class 13/51 images, making it so that the network is *encouraged* to learn to get those few examples right.

- **Two-step training**: Instead of feeding every class straight into a network, a process is run to try and differentiate between 3 classes: 53, 13, and every other class. This way, the cardinality of each class is roughly the same. A second CNN can be used to classify the "other class" as a specific class from the labels provided.

Having defined the architecture and strategies to deal with the class imbalance, we first look at the results of the first, 3-class CNN. In terms of metrics, we monitored both the accuracy and loss over-time, and in the class of the 48-class classifier, we also used top-3 and top-5 accuracy. At the end of training, we also pass a test set through the network, and calculate the confusion matrix and calculate some metrics, such as precision, recall and f1-score. These metrics are important in helping us analyse the network's behavior, for example if the network is over-classifying the images as one specific image, or if it is completely ignoring a class altogether.

After training the initial 3-class CNN, we obtain the following results:

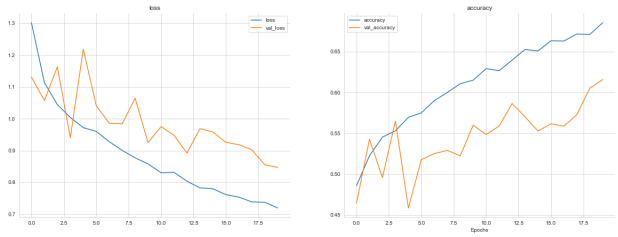

Figure 7: 3-class CNN's Training metric graphs (from left: loss, accuracy)

$$\begin{array}{r} \textit{Class } 51 \\ \textit{Class } 13 \\ \textit{Others} \end{array} \begin{bmatrix} 299 & 39 & 153 \\ 78 & 379 & 431 \\ 180 & 22 & 516 \end{bmatrix}$$

Figure 8: 3-class CNN's confusion matrix

	Precision	Recall	f1-score	#
-1	0.54	0.61	0.57	491
13	0.86	0.43	0.57	888
51	0.47	0.72	0.57	718
macro avg	0.62	0.58	0.57	2097
weighted avg	0.65	0.57	0.57	2097

Figure 9: 3-class CNN Testing Metrics

From the accuracy and loss curves, we can see some ups and downs at the end, but eventually the model starts to converge. This may be explained due to the use of Dropout layers, which randomly resets connections between dense layers, causing the network to take longer to converge to stability.

From the confusion matrix and the precision/recall scores, we can gather a few points:

- The network seems to be good at identifying class 13, with 86% precision. However, the 42% recall value tells us it is also classifying a lot of images from class 13 as something else, namely class 51, which takes us to our next point;
- The network seems to be over-classifying images as class 51; from the confusion matrix, we can see that images from classes -1 and 13 are often labeled as class 51, and in the case of class 13, even more times than the number of times it guesses correctly.

Moving on to the 48-class CNN, after training we obtain the following results:

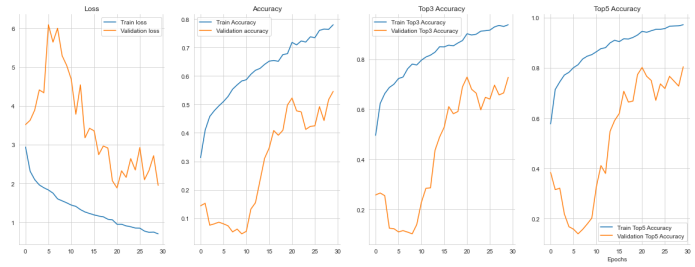

Figure 10: 48-class CNN's Training metric graphs (from left: loss, accuracy, top-3 accuracy, top-5 accuracy)

	Precision	Recall	f1-score	#
macro avg	0.31	0.28	0.27	297
weighted avg	0.63	0.66	0.62	297

Figure 11: 48-class CNN Testing Metrics

We cannot show the confusion matrix since its size is so large it would be almost incomprehensible (48x48), but the testing metrics do give us some insights:

- The macro averages are much lower than the weighted averages, and this is because there are many classes that have very low representation in the dataset (<20 samples), which the model does not learn to predict correctly, and as such result in 0% precision, recalls and f1-scores.
- The reason the weighted averages are so high is because the classes with the highest precision, recall and f1-score values are, naturally, the classes that appear the most in the dataset; the network has trained to recognise these classes better, and, as such, these classes return better metrics, which are weighted more due to their dominance.

Putting the two networks together, the following metrics are obtained:

	Precision	Recall	f1-score	#
macro avg	0.03	0.02	0.02	2097
weighted avg	0.41	0.36	0.27	2097

Figure 12: Final Testing Metrics combining both CNNs

Overall, the performance of the CNNs could be better, but due to the great class imbalance of the dataset, and the existence of labels with so little as 10 images for training, these are not unexpected results.

2.2 Multi-label classification

After being able to classify the images as a single class, a different problem was considered: predicting a set of labels associated with each image.

For this task, a similar architecture to the one used in Approach 2 was considered, based on an AlexNet Convolutional Neural Network. The main difference is in the activation function, as we used **sigmoid** instead of softmax. This change is important because a softmax activation enforces that the sum of the probabilities of the output classes is equal to one, so in order to increase the probability of a particular class, the model must decrease the probability of other. As in this case we want to be able to predict more than 1 label for each image, a sigmoid function, that does not make the outputs mutually exclusive, is more adequate.

We trained the network on 2 datasets of different sizes, whose results are presented below:

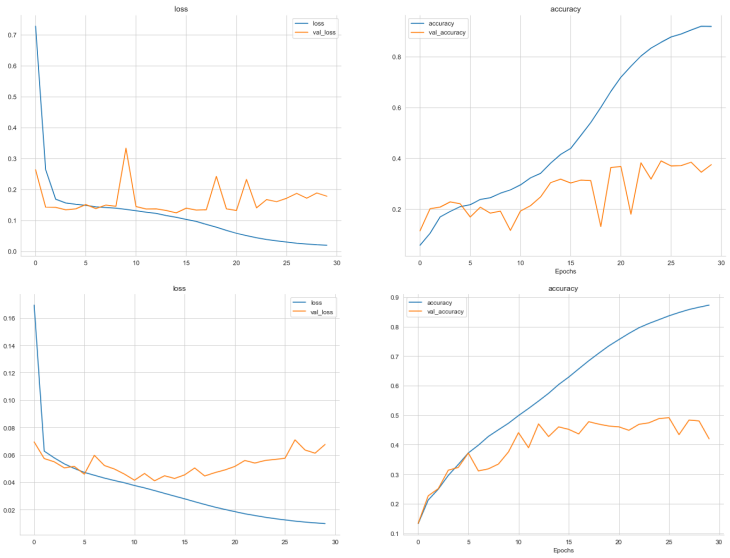

Figure 13: Multilabel CNN's Training metrics graphs (small dataset above, large dataset below)

We can notice a considerable improvement of loss and accuracy when we trained on the larger dataset. The loss went from 0,2 to 0,06 and the accuracy from 0,4 to 0,5. However, in both cases we can see that the

network stops improving at some point, when it starts to overfit. Even though many techniques were used to prevent this from happening, such as Dropout layers, shuffling data, Batch normalization and pixel normalization, we consider that this happens due to there not being enough data for the model to train.

For evaluation of the results, we used the F2-score metric, which is one of the Fbeta-measures, where beta=2. This metric is similar to F1-score, but it puts more weight on recall than it does on precision, which is to say it values finding every relevant image of the class more than it values not flagging irrelevant images as the specified class.

Here we can see a table with these results:

	f2-score
micro avg	0.26997
macro avg	0.07253
weighted avg	0.25374

Figure 14: Final Testing Metrics of Multi-label CNN

Similarly to the first task, macro average suffers due to there being quite a few classes with little training images and, consequently, very poor performance in the CNN. However, the micro avg for the f2-score, which is more suited for imbalanced datasets, reveals that the network has a much better accuracy for those models that are better represented in the dataset. The accuracy obtained seems comparatively worse than task 1's, but it must be remembered that the problem is one of higher complexity.

3 Considerations

3.1 Performance

Over the course of this work, we had quite a bit of trouble with the performance of the systems, especially when it came to training time; since none of us had a particularly powerful system, and the Google Collab tools seemed to freeze up and even crash at some points, training the models proved to be quite a challenge. However, we managed to train models that were enough to discuss metrics and analyse results, which was the main objective of this work. Although some of the results revealed our solutions to be somewhat inaccurate, we consider the metrics and methodologies used to train and study these models to have been the most adequate. It would probably be possible to obtain better results by using a better network architecture, such as a VGG-16 [4], but due to hardware limitations, we were forced to take a simpler approach.

3.2 Problems

Due to some imbalances in class representation in some datasets, especially in the first task, we found avoiding over-fitting a big problem. Fortunately, with the use of some techniques such as normalization, data augmentation and dropout layers, we were able to mitigate this problem to a certain level, but it was never completely overcome.

4 Conclusion

We successfully developed a system that is capable of classifying and attributing labels to artwork images from the collection of The Met. Different state of the art approaches were explored and analysed in detail in this article. From this experience, Convolutional Neural Networks have proven to be a reliable means to implement such a system, with a lot of applications, even though they require a considerable amount of time to train. As future work, different types of deep neural networks' architectures, apart from AlexNet, could be analysed, so as to understand their suitability to this scenario. Additionally, other, more balanced, datasets could be explored to better measure the performance of the system.

References

[1] *C-Support Vector Classification model class, from the Scikit-Learn Python library.* URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[2] Hinton G. Krizhevsky A. Sutskever I. *ImageNet Classification with Deep Convolutional Neural Networks.* URL: https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68Paper.pdf. (accessed: 01.06.2021).

[3] Khoshgoftaar T. Shorten C. *A survey on Image Data Augmentation for Deep Learning.* URL: https://journalofbigdata.springeropen.com/track/pdf/10.1186/s40537-019-0197-0.pdf. (accessed: 03.06.2021).

[4] Zisserman A. Simonyan K. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* URL: https://arxiv.org/pdf/1409.1556v6.pdf. (accessed: 01.06.2021).