Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Final Project

Image orientation detection

Authors:
Vasco Coelho - 807304 - v.coelho@campus.unimib.it
Lorenzo Mauri - 807306 - l.mauri28@campus.unimib.it

June 18, 2021

Contents

1 Introduction 2

2 Datasets 4

3 The Methodological Approach 5

4 Results and Evaluation 8

5 Discussion 10

6 Conclusions 11

Abstract

Image canonical orientation detection requires high-level scene understanding. The vast amount of semantic content in images makes orientation detection challenging. Over the last years, deep convolutional neural networks (ConvNets) have transformed the field of computer vision thanks to their unparalleled capacity to learn high level semantic image features. Similar to human perception, ConvNets can exploit implicit object recognition and other useful semantic cues to predict the correct orientation of an image. In this work, the power of deep learning has been exploited and a pre-trained ConvNet adapted for the image orientation detection task. An evaluation of the constructed ConvNet shows that it greatly generalizes to correctly orient a large set of test images reaching a 95% of accuracy, which is close to that of humans. It allows to disregard meta-data such as exchangeable image file format (EXIF) which may not be consistently stored or manipulated across different devices, applications, and file formats.

1 Introduction

Nowadays, several image processing systems require correctly oriented images before processing and visualization. For instance, state-of-the-art convolutional neural networks usually achieve optimal performance when images are in the upright orientation for tasks such as classification and retrieval. Furthermore, modern cameras and smartphones write orientation tags in the exchangeable image file format (EXIF) [1] field in the image meta-data. However, this field is not consistently recorded, manipulated and updated across different software applications, therefore the image may be displayed in its original, unrotated state. While rotation by any angle is in principle possible, rotations by multiple of 90 degrees are more common, and are also straightforward to correct once detected. Owing to the wide variety of scene content, automatically detecting the right orientation is a challenging task.

Orientation detection requires high-level visual content understanding, especially when dealing with indoor scenes. The main difficulty is that while some indoor scenes (e.g. corridors) could be well characterized by global

[1] The metadata tags defined in the EXIF standard cover a broad spectrum of informations: date and time, camera settings, descriptions, copyright informmation, etc. Further details are available at https://it.wikipedia.org/wiki/Exchangeable_image_file_format

spatial properties, others (e.g., bookstores) are better characterized by the objects they contain.

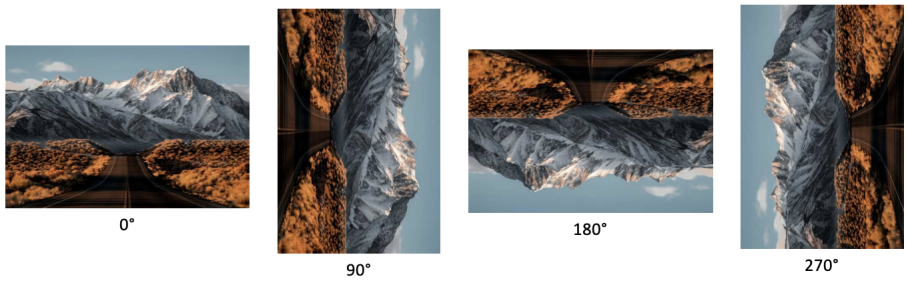

Figure 1: The four canonical orientations of an example image are shown: 0, 90, 180 and 270 degrees.

When it comes to images, convolutional neural networks (CNNs, or ConvNets) are a promising technique for feature extraction, as they can take into account both low-level and high-level features. This study is focused on providing a solution, based on CNNs, to find the correct canonical orientation of an image (Figure 1). The architecture of the *VGG16* convolutional neural network was initially considered as a starting point, given the exceptional 98.5% accuracy achieved by *U.Joshi et. al* [1] on the SUN397 dataset, although the *AlexNet* architecture has been used with excellent results by *K.Swami et. al* [2].

Since there were no benchmark datasets for the namely task, it was manually generated for this project. To suitably generalize across different image domains, this dataset comprises a partition of the *Scene UNderstanding (SUN)* [2] and a subset of the *Indoor Scene Recognition* [3] database, containing a large number of both indoor and outdoor scenes altogether.

[2] https://web.mit.edu/torralba/www/indoor.html
[3] https://vision.princeton.edu/projects/2010/SUN/

2 Datasets

This section focuses on the properties of the datasets, considered in this project and explains how the data was prepared for training. One of the two main data sources was the large-scaled benchmark dataset for scene recognition **SUN397** [3] (similar to Ciocca et al. [4]). It contains 108754 images in 397 different categories with at least 100 images per category. More precisely, the first of 10 official partitions of the dataset was used, which originally involves 39700 distinct images. This partition has 50 images per category both for training and testing. The remaining data source is the MIT **Indoor** dataset [5], containing 15620 images in 67 different categories with at least 100 images per category. This is because it includes a significant number of background clutter and lacks discriminative features. A selected subset of the Indoor dataset primarly comprises 6661 distinct images, consisting of 80 training images and 20 testing images per category.

It is worth mentioning, training, validation and test sets were balanced by rotating each image by 90, 180 and 270 degrees. This strategy is cost-effective and makes it possible to generate labels for free for both SUN397 and Indoor datasets. In general, it could be applied without any prior labeling when the number of misoriented images is small compared to the total size of the dataset. After generating a copy of each image in canonical orientations, 158800 and 26644 images corresponding to the first partition of SUN397 and a subset of the Indoor dataset, respectively, were retrieved. As a result, the analysis included 185444 images, which involved the first partition of the SUN397 and a subset of Indoor dataset. Training, validation, and testing datasets were randomly extracted from the SUN397 and Indoor datasets.

Table 1: Number of generated images after the dataset *generation step*.

	Train	**Validation**	**Test**	Total
SUN397	63520	15880	79400	158800
Indoor	17034	4258	5352	26644
Total	80554	20138	84752	**185444**

During the dataset generation phase each image was resized to 224 by 224 pixels and converted to have three channels, not only because of the default input of the selected architecture but also to boost the data transfer from *Google Drive* to *Google Colab*.

Thanks to its excellent downscaling and upscaling quality, *cubic interpolation* method was applied on rescaled images. However, it computationally is more expensive than linear interpolation methods, thus the scaling was performed along with the dataset *generation step*. Finally, a balanced dataset was generated, aiming at eliminating the effect of any bias.

The *Colab* notebook that implemented the dataset *generation step* is available on Google Drive at the following link *generator.ipynb*.

3 The Methodological Approach

An entire convolutional neural network (CNN, or ConvNet) has insufficiently been trained from scratch with random initialization, since it is relatively rare to have a dataset of appropriate size. Instead, it is common to pre-train a CNN on a very large dataset (e.g. *ImageNet*, containing 1.2 millions of images with 1000 categories), and then use the CNN either as a starting point to customize or as a fixed feature extractor for the task of interest. It should be noted that ConvNet features are more generic and original-dataset-specific in early and later layers, respectively. That said, *transfer learning* was performed from the *VGG16* ConvNet [6], previously trained on the large scale *ImageNet* classification task. This was motivated by the observation that the *earlier* layers of a ConvNet contain more generic features, like edge detectors or color blob detectors, which should be useful for many tasks. In contrast, *later* layers of the ConvNet becomes progressively more specific to the details of classes contained in the original dataset. The intuition behind transfer learning for image classification was that if a model is trained on a large and general enough dataset, it will effectively be served as a generic model of the visual world. The structure of the *VGG16* CNN was made up of five *convolution* blocks and 3 *fully connected* layers for classification. The first two blocks included two *convolutional* layers. The rest of them had three convolutional layers each. All convolution blocks were followed by one *max pooling* layer. All convolutional layer filters had a very small receptive field (3×3) which is the smallest size to capture the notion of left/right, up/down, center.

The first training attempt - using the original *VGG16* as a fixed feature extractor - did not show promising results. Simply replacing and training from the very beginning with random initialization, the *head* classifier - formerly containing the 1000 class scores of the *ImageNet* task and subsequently

only containing the 4 class scores related to the orientation detection task - was not enough (84% of accuracy on the validation set). Therefore, the network has undergone some changes, some of these similar to the work of Morra et al. [7]. Moreover, dropout and batch normalization layers have been introduced.

A new strategy was based on doing more than just the extraction from a previous network of meaningful features from new images. A few of the top layers of the frozen model base were unfreezed and further trained by continuing the backpropagation up to and including the fifth convolution block. This second attempt, consisting of jointly training both the newly-added classifier layer and the last layers of the base model allowed to *fine tune* the higher-order feature representations to make them more relevant for the task under analysis. The first four convolution blocks had been kept fixed and only the higher-level portion of the network was fine-tuned. As a result, $< 1\%$ and approximately 94% of the total parameters of the *VGG16* base model were trained in the first and second training phase, respectively.

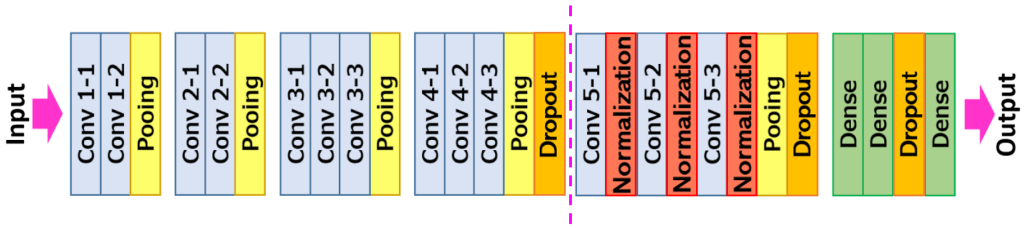

Figure 2: The final custom ConvNet built from the original *VGG16* architecture. The dashed line in pink indicates the cut from which the *fine tuning* was performed.

The original structure of the *VGG16* model mostly was retained, except for the head classifier which has now 4 outputs. To reduce the amount of overfitting, three dropout layers were added with the probability of 0.15, 0.15, 0.25, respectively, after the fourth convolution block, after the fifth convolution block, and after the second fully connected layer; and finally batch normalization [8] layers were added following each convolutional layer in the fifth convolution block, as depicted in Figure 2.

Both training experiments were performed using a batch size of 64, the Adam optimizer [9], the *sparse categorical crossentropy* loss function (to compute the crossentropy loss between labels and predictions) and the *accuracy*

metric (to calculate how often predictions equaled labels). The initial learning rate was 1×10^{-4} in the *feature extractor experiment*, which lasted 13 epochs with early stopping using 2 epochs of patience. The *fine tune experiment* lasted 3 epochs during which a lower learning rate (1×10^{-6}) was used otherwise the model could have overfitted very quickly. The second experiment was done shortly after the first one, that is, after that the top-level classifier and the freezed pre-trained *VGG16* model converged. Adding a randomly initialized classifier on top of a pre-trained model and attempting to jointly train all layers could make the magnitude of the gradient updates too large (due to the random weights from the head classifier), thus forget what the model knew.

Therefore, some post-training quantization techniques were applied to reduce model size while also improving CPU and hardware accelerator latency, with little degradation in model accuracy.

The *Colab* notebook that implemented the *training step* is available on Google Drive at the following link *trainer.ipynb*.

4 Results and Evaluation

All experiments were performed using *TensorFlow* deep learning framework on *Google Colaboratory* with GPU support. The second phase of the analysis, corresponding to the *training step* has been timed resulting in a duration of roughly 2 hours and 18 minutes.

Before training, the initial accuracy on the validation set in the *feature extractor experiment* was about 26.6% with a 2.79 loss. After ten epochs, each lasting about 8 minutes, the results reported in the first row of Table 2 were achieved. Learning curves obtained in the two experiments are reported in Figure 3. Validation metrics were clearly better than the training metrics because *batch normalization* and *dropout* layers affected accuracy during training. They were turned off when calculating validation loss. To a lesser extent, it was also because training metrics reported the average for an epoch, while validation metrics were evaluated after the epoch, so validation metrics saw a model that had trained slightly longer. While the accuracy score is only a global measure, it was not sufficient to fully evaluate the model's predictions to understand if the learning procedure was reliable or not. Finally, rigorous evaluation of ConvNet's performance was carried out creating the confusion matrix corresponding to the test set in the *fine tune experiment*, reported in Figure 4.

For evaluation, all images were initially in the correct orientation and randomly selected images were rotated by 90, 180 or 270 to generate a balanced testing set. The most probable class was taken as the predicted class. Results show a remarkable overall 95% accuracy on unseen randomly rotated images, which is close to human performance and in line with previous results in literature [4]. Different results may be attributed partly to the different training, validation and test split, since there is no official benchmark for this specific task.

Table 2: Accuracy obtained both in the *feature extractor* and in the *fine tune* experiments on training, validation and test set respectively.

Accuracy	**Training Set**	**Validation Set**	**Test set**
feature extractor experiment	81.62%	84.25%	84.45%
fine tune experiment	94.18%	93.42%	95.77%

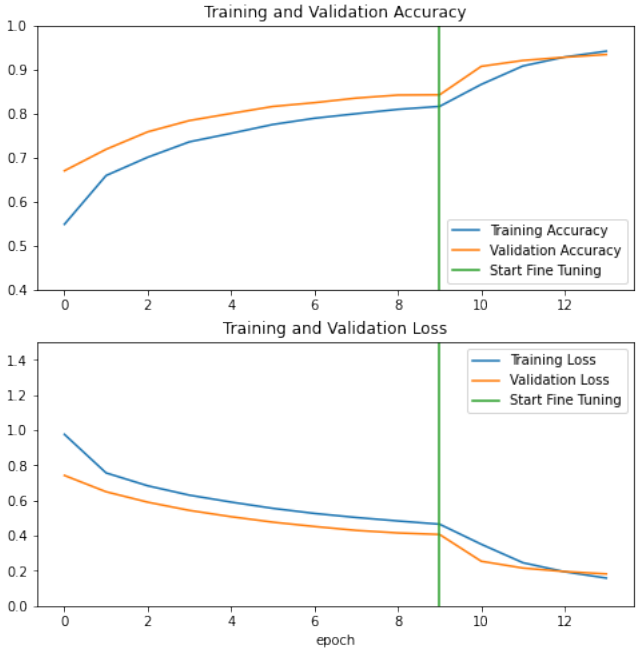

Figure 3: Learning curves obtained during the *training step*, describing all 13 epochs. The vertical green line indicates when the *fine tuning* began.

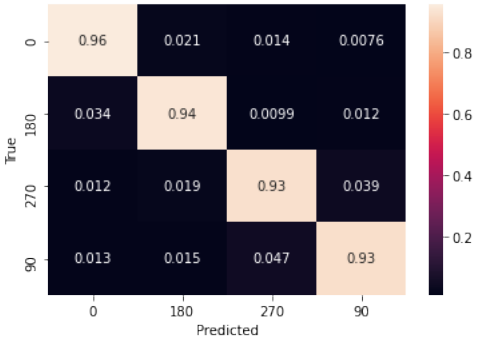

Figure 4: Confusion matrix reporting the percentage of correct (main diagonal) and incorrect classifications for each canonical orientation.

5 Discussion

In light of the project's objectives, the results obtained have shown unpredictable success: the final model achieved an accuracy of more than 95%, which is very close to that of humans. Hence, from a practical point of view, it has been proven once again that reusing or transferring information from previously learned tasks to learning new tasks has the potential to significantly reduce the amount of computational resources needed if starting from scratch. It took only a few hours to fully train the final proposed ConvNet. Actually, the most time-consuming part was the dataset *generation step*, in which a large amount of data has been processed and then moved from *Google Drive* to *Google Colaboratory* and vice versa, as the whole analysis included 185444 images totally. From the investigation of the performance measures relating to the first *feature extractor experiment*, a noteworthy aspect emerged: simply by instantiating the *VGG16* pre-trained model and adding a fully-connected classifier on top of it and some dropout and batch normalization layers from the fifth convolution block on, an accuracy of 84% was reached as shown in Table 2. This implies that the orientation detection task and the *Imagenet* one, that the pre-trained model was trained on, are not that different after all. The pre-trained model was frozen and only the weights of the head classifier got updated during training to achieve the previously-mentioned accuracy. In Figure 3 it is possible to see how the slope of the learning curves, that is the speed of convergence over epochs, changed due to the start of the *fine tune experiment* between the ninth and tenth epochs. The *training step* lasted 13 epochs and no longer to avoid a possible over-learning of the model on the training data that would have made the model completely useless. In Figure 4, the confusion matrix highlights the fact that it is more difficult for the ConvNet to correctly predict counterclockwise rotations of 90 and 270 degrees than those of 0 and 180.

Based on the results of the whole analysis, perhaps the experimental procedure can be improved by using a pre-trained architecture other than *VGG16* as a starting point. Some work has already been done in this regard and a selection of ConvNets has been made from the *Tensorflow Hub* repository of trained machine learning models ready for fine-tuning. The *Colab* notebook that contains the possible future development just mentioned is available on Google Drive at the following link *future_development.ipynb*.

6 Conclusions

The main goal of this project was to provide a model capable of detecting the correct orientation of images. This objective was achieved by fine-tuning the well known *VGG16* ConvNet pre-trained on *Imagenet*. An accuracy of 95% was achieved, which is close to that of humans, on a test set containing nearly 85 thousand images specifically built for the task under consideration. The proposed approach, providing an accurate deep learning algorithm, demonstrates the effectiveness of the computer vision technique called transfer learning. Future developments will delve into other pre-trained architectures to achieve better performances while maintaining good generalization across different image domains.

References

[1] U. Joshi and M. Guerzhoy, "Automatic photo orientation detection with convolutional neural networks," in *2017 14th Conference on Computer and Robot Vision (CRV)*, 2017, pp. 103–108.

[2] K. Swami, P. P. Deshpande, G. Khandelwal, and A. Vijayvargiya, "Why my photos look sideways or upside down? detecting canonical orientation of images using convolutional neural networks," in *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 2017, pp. 495–500.

[3] J. Xiao, J. Hays, K. A. Ehinger, A. Oliv, and A. Torralba, "Sun database: Large-scale scene recognition from Abbey to Zoo," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*, p. 3485–3492, 2010. [Online]. Available: https://vision.princeton.edu/projects/2010/SUN/paper.pdf

[4] G. Ciocca, C. Cusano, and R. Schettini, "Image orientation detection using lbp-based features and logistic regression," *Multimedia Tools and Applications*, vol. 74, no. 9, p. 3013–3034, 2015. [Online]. Available: https://doi.org/10.1007/s11042-013-1766-4

[5] A. Quattoni and A. Torralba, "Recognizing indoor scenes," *IEEE Conference on Computer Vision and Pattern Recognition*, p. 413–420,

2009. [Online]. Available: http://web.mit.edu/torralba/www/indoor.html

[6] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2015.

[7] L. Morra, S. Famouri, and F. Lamberti, "Automatic detection of canonical image orientation by convolutional neural networks," *IEEE 23RD International Symposium on Consumer Technologies (ISCT 2019)*, 2019. [Online]. Available: http://dx.doi.org/10.1109/ISCE.2019.8901005

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.