Università degli studi di Milano-Bicocca

Advanced Machine Learning

Final Project

Image orientation detection

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

June 17, 2021

Contents

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 very 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

The problem addressed in this project has attracted researchers for decades, as devices have always had trouble recording metadata, including orientation tags. Modern cameras or smartphones, in fact, write orientation tags in exchangeable image file format (EXIF) [1] structures, which is extremely useful in some cases. For example devices may not update their metadata when photos are manually rotated or certain applications may not support EXIF tags. In other cases they may even delete them (i.e while saving images) or may have sensors that do not record tags correctly (i.e it is the case of sports cameras). For these reasons, a deep learning approach could represent a valid way to support existing techniques and perhaps to provide new solutions.

Canonical 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) can be well characterized by global spatial properties, others (e.g., bookstores) are better characterized by the objects they contain. Since our model must be capable of generalizing

[1] Containing data like image resolution, camera used, color type, compression as well as orientation tags, etc. Further details are available at https://it.wikipedia.org/wiki/Exchangeable_image_file_format

well on new images, both scenarios have been take into account considering therefore two scene recognition dataset. More specifically the dataset contains a partition of the **Scene UNderstanding (SUN) database** [2] and a subset of the **Indoor Scene Recognition database** [3].

As stated previously we focused our attention at learning a model to classify an input image into the four classes shown in Figure 1.

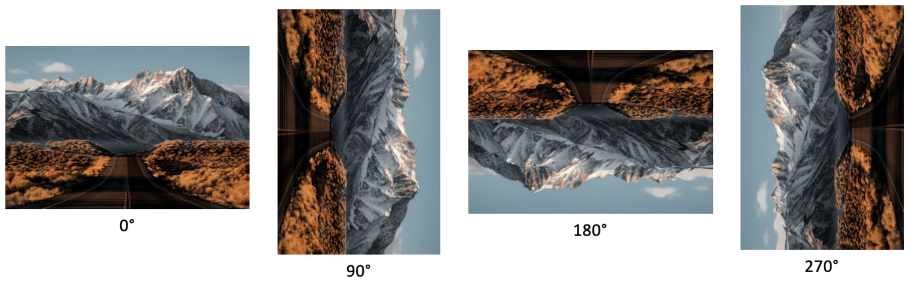

Figure 1: An input image with correct labels. The possible output values are 0°, 90°, 180° and 270°

The classifier consists of a Convolutional Neural Network (CNN) whose architecture is a modified VGG-16, a well known model originally designed for the ImageNet challenge. Finally, we used a common practice called *Transfer Learning* that relies on adapting a pre-trained neural network to a new but similar domain respect the original one.

1.1 Related Work

As far as the dataset generation is concerned, *M. Prince et al.* [1] used both outdoor and indoor images for a similar task as well as *L.Morra et. al* [2]. They achieved in their works accuracies of 95.23% and 96.82% respectively, outperforming the state-of-the-art.

Despite AlexNet was employed with excellent results, we decided to use VGG-16 given the higher accuracies of 98.5% reached by *U.Joshi et. al* [3] on SUN397 dataset.

[2] http://groups.csail.mit.edu/vision/LabelMe/NewImages/
[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 considered is the large-scale benchmark dataset for scene recognition **SUN397** [4] (similar to Ciocca et al. [5]). It contains 108754 images in 397 different categories with at least 100 images per category. To be precise, the first of the 10 official partitions of the dataset was used in the analysis performed, which involves originally 39700 distinct images. This partition has 50 training images and 50 testing images per category. The other main data source considered is the MIT **Indoor** dataset [6] which contains 15620 images in 67 different categories with at least 100 images per category. This dataset was chosen because it contains lots of background clutter and lack discriminative features. The subset of the Indoor dataset actually used contains originally 6661 distinct images. This subset consists of 80 training images and 20 testing images per category.

For training, validation and test a balanced training set was generated by rotating each image by 90, 180 and 270 degrees. This counterbalances the lack of data for less frequent classes, such as images rotated by 180. This strategy is cost-effective and makes it possible to generate labels for free for both SUN397 and Indoor datasets, and in general can be applied without any prior labeling when the amount of misrotated images is small compared to the total size of the dataset.

Consequently, after generating a copy of each image in the four canonical orientations, 158800 images corresponding to the first partition of SUN397 is obtained and 26644 images corresponding to the subset of the Indoor dataset is retrieved as well. As a result, the analysis includes 185444 images totally, which involves the first partition of the SUN397 dataset and the a subset of the Indoor dataset. Training, validation, and testing datasets were randomly extracted from the SUN397 and Indoor datasets.

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

	Train	**Validation**	**Test**	Total
SUN397	63520	15880	79400	158800
Indoor	17034	4258	5352	26644
Total	80554	20138	84752	**175444**

During the dataset generation phase each image was resized by $224x224$ pixels both because this is the default input size for the selected architecture and to speed up the data transfer from *Google Drive* to *Google Colab*. Images have been resized employing a *cubic interpolation* method considering its good downscaling and upscaling quality. Despite this, it is computationally more expensive than linear interpolation methods so it was decided to do the scaling along with the dataset generation phase.

Finally, a balanced dataset was generated, i.e., equal number of images for each orientation category to eliminate the effect of any bias.

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

3 The Methodological Approach

Nowadays, very few people train an entire convolutional neural network (CNN, or ConvNet) from scratch with random initialization, because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pre-train a CNN on a very large dataset (e.g. *ImageNet*, which contains 1.2 million 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. Keeping in mind that ConvNet features are more generic in early layers and more original-dataset-specific in later layers, *transfer learning* was performed from the *VGG16* ConvNet [7] previously trained on the large scale *ImageNet* classification task. This was motivated, as stated previously, by the observation that the *earlier* layers of a ConvNet contain more generic features like edge detectors or color blob detectors that should be useful to many tasks, but *later* layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset. The intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. The structure of the *VGG16* CNN is made up of 5 *convolution* blocks and 3 *fully connected* layers for classification. The first two blocks are made up of two *convolutional* layers. The remaining three convolution blocks have 3 convolutional layers each. All convolution blocks are followed by one *max pooling* layer. All convolutional layer filters have 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 scratch the *head* classifier, formerly containing the 1000 class scores for the *ImageNet* task and now 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. [8]. Moreover, dropout and batch normalization layers have been introduced.

A new strategy was adopted, trying to do more than just extracting meaningful features from new images using representations learned by a previous network. 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 in order to make them more relevant for the task under analysis. Consequently, the first four convolution blocks have been kept fixed and only the higher-level portion of the network has been fine-tuned. As a result, $< 1\%$ and approximately 94% of the total parameters of the *VGG16* based model were trained in the first and second training phase respectively.

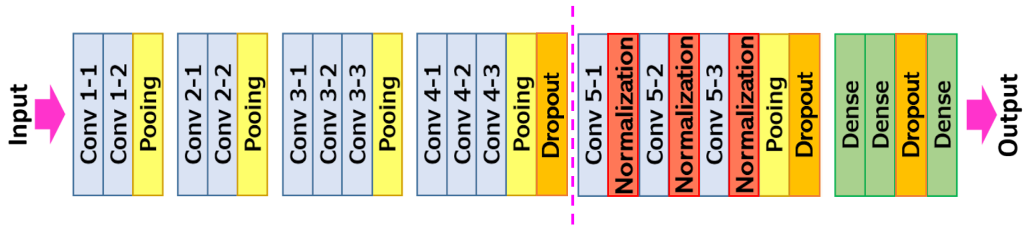

Figure 2: The final custom ConvNet built from the original *VGG16* architecture. The dashed line in pink indicates the cut from which the *fine tuning* was performed.

The original structure of the *VGG16* model was mostly kept, except for the head classifier which has now 4 outputs. To reduce the amount of overfitting, three dropout layers were added with probability equal to 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 [9] layers were added after each convolutional layer in the fifth convolution block, as shown in Figure 2.

Both training experiments were performed using a batch size of 64, the Adam optimizer [10], the *sparse categorical crossentropy* 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. This second experiment was attempted after the first one, that is after having trained to convergence the top-level classifier with the pre-trained *VGG16* model freezed. Adding a randomly initialized classifier on top of a pre-trained model and attempting to train all layers jointly could have made the magnitude of the gradient updates too large (due to the random weights from the head classifier) and therefore forget what the model knew.

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

The *Colab* notebook that implements the *training step* is available on Google Drive at the following link *vgg16.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 approximately 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. Finally, the confusion matrix corresponding to the test set in the *fine tune experiment* is 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 [5]. 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%

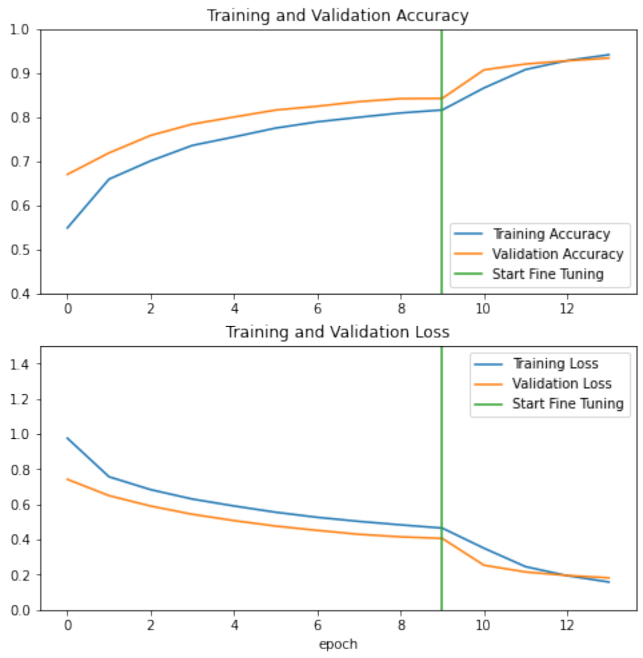

Figure 3: Learning curves obtained during the *training step*, describing all 13 epochs. The vertical green line indicates when the *fine tuning* began.

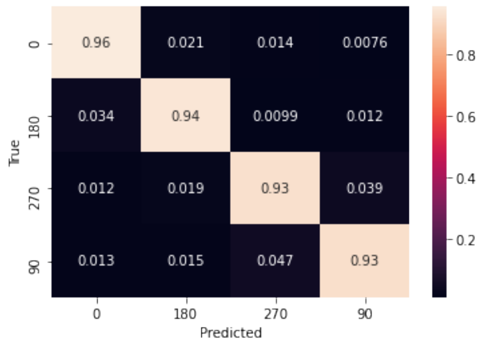

Figure 4: Confusion matrix reporting the percentage of correct (main diagonal) and incorrect classifications for each canonical orientation.

5 Discussion

To be noteworthy, an advanced predictive model must be accurate on unseen data. Sometimes, indeed, a not uncommon phenomenon called *overfitting* occurs because the model has just stored patterns belonging to the training data but has failed to generalize the results on new data. Unfortunately this scenario renders the model completely useless. For this reason a rigorous evaluation is fundamental in order to understand if the learning procedure is reliable or not. In this regard, graphical representations of statistical measures can be considered to evaluate the performances.

5.1 Overall Results

5.1.1 Accuracy and Confusion Matrix

As far as the specific case is concerned,an initial analysis suggests that the classifier can solve the task in a quite satisfying way since the validation and test accuracies are approximately equal to 93% and 96% as shown in Table 2. However analyzing the learning and loss curve trends over epochs and having a deeper detail on classification accuracy for each class may be of interest.

On the other hand, the accuracy score itself is just a global measure and it is not enough to fully assess the model predictions. Indeed, while the accuracy is effective to get the hang of the general performances , it does not provide further informations about each category. As a result, creating a *confusion matrix* is a common practice that allows to go deeper in the analysis.
The confusion matrix (or error matrix) is a specific table layout that allows visualization of the performance of an algorithm where each row represents the instances in an actual class and each column represents the instances in a predicted class, or vice versa [4]. In particular, having four categories implies a 4x4 matrix.
It is possible to note that the fine-tuned model shows good performances on every class (Figure) while the Head VGG-16 model shows some difficulties at classifying instances belonging to the 90 degrees orientation group (Figure).

[4]https://en.wikipedia.org/wiki/Confusion_matrix

5.2 Training and Validation History

As stated previously, seeing all the whole learning process may be of interest, especially to assess the effectiveness of the applied methods. The Figure allows to understand and visualize some properties of the VGG-16 head classifier, that are :

- The speed of convergence over epochs (slope)
- The convergence of the model (plateau of the line).
- The possible over-learning of the model on the training data (inflection for validation line)

while the Figure shows the point where the *fine-tuning* is applied and the consequent changes.

In this scenario, the model increases its performance after the tenth epoch in which it passes from just having one trainable layer to a major flexibility : the training process therefore forces weights of the model to change from generic feature maps to features associated specifically with the dataset.

6 Conclusions

The main goal of this project was to provide a model to correctly classify the image orientation into four coarse angles of 0, 90, 180 and 270 degrees.

The proposed approach, which surely provides an accurate deep learning algorithm, demonstrates also the effectiveness of the Computer Vision technique called Transfer Learning and, in particular, the Fine Tuning one. Indeed, the last one was fundamental to obtain better performances respect with those of the *head model*.

As stated in the Introduction (Section) one could be interested on deploying the algorithm onto a mobile device. In this regard, a suggested enhancement could be to compress the size of the ConvNet or to take into account other lighter Deep Learning architectures (e.g MobileNet which has less in parameters). In addition, an application was developed in a related work to enable ordinary users to easily leverage such an advanced predictive model for image canonical orientation detection.

References

[1] M. Prince, S. A. Alsuhibany, and N. A. Siddiqi, "A step towards the optimal estimation of image orientation," *IEEE Access*, vol. 7, pp. 185 750–185 759, 2019.

[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," *CoRR*, vol. abs/1712.01195, 2017. [Online]. Available: http://arxiv.org/abs/1712.01195

[3] U. Joshi and M. Guerzhoy, "Automatic photo orientation detection with convolutional neural networks," in *Proceedings - 2017 14th Conference on Computer and Robot Vision, CRV 2017*, ser. Proceedings - 2017 14th Conference on Computer and Robot Vision, CRV 2017. United States: Institute of Electrical and Electronics Engineers Inc., Feb. 2018, pp. 103–108, 14th Conference on Computer and Robot Vision, CRV 2017 ; Conference date: 17-05-2017 Through 19-05-2017.

[4] 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

[5] 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

[6] 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

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

[8] 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

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

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