AN2DL First Homework

Mauro Famà, Sofia Martellozzo, Lorenzo Mondo

Group cANNoli

Academic Year 2022-2023

1 Introduction

This report wants to explain our methodologies and experiments performed during the image classification task over a dataset of plants using Convolutional Neural Networks models. The dataset we have worked with is composed of 3542 images of plants divided in 8 different classes. Because of the small size of the dataset, our main experiments consist of transfer learning and data augmentation techniques. Into the following section we'll describe our work and the decisions that led us to the CNN model that performed the best accuracy result (0.86%) over the test dataset.

2 Techniques

2.1 Data Augmentation

It is common knowledge that the more data an ML algorithm has access to, the more effective it can be[0]. Even when the data is of lower quality, algorithms can actually perform better, as long as useful data can be extracted by the model from the original data set.
In this homework we are required to classify species of plants, the given dataset is significantly smaller than the datasets used to train famous Convolutional Neural Networks (CNN), so we decided to resort to Data Augmentation. A CNN is translation invariant but not rotation invariant[1]. The plants, however, had no fixed orientation. We can therefore generate more training data by rotating the original training data. As plants were assumed to be symmetric, even more training data can be generated by mirroring the training data. The training set was thereby increased by mirroring the images and rotating them in 90° increments.

2.2 Class Weights

Imbalanced classification refers to the problem that one class contains a much smaller number of samples than the others in classification. In deep learning, it is important to train an estimator on balanced data so the model is equally informed on all classes. The Scikit-Learn classifiers have a `class_weights` parameter that can be set to declare how to rank the importance of imbalanced data. Instead of actually oversampling to balance the classes, we can inform the estimator to adjust how it calculates loss.

2.3 Transfer Learning and Fine Tuning

Transfer learning is the process to take a network model that has already been trained for a given task and make it perform a second similar task. The pre-trained models we used are part of the Keras Application library, with their implementations and pre-trained weights on a common dataset like ImageNet.
We replaced the output layer, originally trained to recognize (in the case of Imagenet models) 1,000 classes, with a layer that recognizes the 8 classes we needed. The new output layer, created by us, is attached to the model and is then trained to take the most peculiar features from the front of the network and map them to the desired output classes. Once this has been done, we set other layers in the model as `trainable=True` and we decrease the original learning rate so that in further epochs their weights can be fine-tuned for the new task too.

3 Development Models

3.1 From Scratch

We decided to build a model from scratch, by following the best practices currently provided by the State of the Art. In the baseline study we considered[1], a convolutional neural network is desired to determine the species of seedlings. The model presented in the study, in order to find the correct number of layers, takes into account two properties: the filter capacity and coverage of the network[2].
The filter capacity is a measure of how well a filter is able to detect complex structures in images. We built our model by following the methods explained in the paper, so this parameter is respected implicitly by construction.

The coverage for a layer in a convolutional neural network is a measure of how big a part of the input image the layer can "see". The receptive field size of the topmost convolutional layer should be no larger than the image size. Our dataset is composed of smaller images with respect to the dataset used in the study (96x96 instead of 128x128), but the coverage of our topmost convolutional layer is 0.93, respecting the constraint of being less than the whole input picture.
In total, the network had 729.864 learnable parameters, which is rather small compared to millions of parameters of the most famous pre-trained networks. Differently from the study, we initialized our weights with the uniform variance scaling initializer instead of the ImageNet weights, so the training phase required more epochs than the baseline study. We trained our model for 210 epochs, reducing the learning rate every 70 epochs. Even if the model was designed for plant classification, it had an accuracy of only 66,26%.

3.2 AlexNet

The architecture consists of eight layers: five convolutional layers and three fully-connected layers. It is different from the classic convolution neural network because: uses Rectified Linear Units (ReLU) instead of the tanh function (that in terms of training time is much slower), allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU and has overlapping pooling (that makes the model more difficult to overfit). We trained our models using stochastic gradient descent, with a batch size of 63.
Our results show that this architecture was not able to predict correctly many images (it achieves a 40% of accuracy). The main reason could be the fact that this network has been developed and trained with a really big dataset (the known ImageNet) with 1.2 million high-resolution images of 1000 different classes. Our problem consists of a significantly smaller dataset with just 8 labels.

3.3 VGG

Because we were not satisfied with our results, we decide to move to transfer learning.
AlexNet came out in 2012 and it improved the traditional convolutional neural networks, so we can consider VGG as a successor of the AlexNet but it was created by a different group named as Visual Geometry Group at Oxford and hence the name VGG, It carries and uses some ideas from its predecessors and improves on them and uses deep Convolutional neural layers to improve accuracy.
VGG16 is a convolution neural network (CNN) architecture that was used to win ILSVR(Imagenet) competition in 2014. The 16 in VGG16 refers to it has 16 layers that have weights. It is considered to be one of the most excellent vision model architecture till date. The most unique thing about VGG16 is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used the same padding and max-pool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. We built the top layer ourselves, we attach 2 FC (fully connected layers) with 2 dropout layers in between, followed by a softmax for output.
The neurons which are "dropped out" in this way do not contribute to the forward pass and do not participate in backpropagation[3]. So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights. This technique reduces complex co-adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons.
At the beginning we train only the fully connected part, by "freezing" the convolutional layers of the architecture downloaded. Then we tried to unfreeze layers by group and re-train the model with a lower learning rate until all the layers were unfrozen. Another guess was to try to unfreeze all the layers after just one train; but in the end, the model that performs better is the one with all the architecture trainable from the beginning. With this model, we achieve an accuracy of 85% .

Another version of the VGG16 found was the VGG19, with 3 additional convolutional layers for the previous network. Regarding the fully connected part, we added 2 Dense layers with much more neurons, but we remove the dropout layers. As before, the best model with this architecture is the one trained entirely from the beginning.

3.4 Xception

The Xception[4] architecture has 36 convolutional layers forming the feature extraction base of the network. The 36 convolutional layers are structured into 14 modules, all of which have linear residual connections around them, except for the first and last modules. It has nearly the same number of parameters as InceptionV3, but we found out that for this classification problem Xception fits better. For the last levers, we added 2 fully connected and dropout modules alternated. In this case, we first train the network with the convolutional part not trainable, then we perform fine-tuning by unfreezing it entirely and re-train it with a lower learning rate.

4 Final Model

Ensemble can be considered a learning technique where many models are joined to solve a problem. This is done because an ensemble tends to perform better than singles improving the generalization ability. There are different ways to ensemble different models together: we adopt the average ensemble, which consists of constructing the weighted arithmetic mean CNN ensemble model[5]. The pooled outputs of our networks are passed through dense layers with a fixed number of neurons in order to equalize them. In this way, we can compute the average. The result is an output that shares the same dimensionality of each input but that is the average of them. As intuitively understandable, the drawback of this approach can be the loss of information caused by the nature of the average operation. With this technique, we combine together the VGG16, Xception, and VGG19 architecture and we achieve an accuracy (in the final part of the challenge) of 86%.

5 Conclusions

We find out that ensemble together different models lead us to a model with higher accuracy. The tree models that we combined perform slightly differently on their own. Another experiment could be to weigh the output of each network, based on their individual performance, when calculating the average result of the ensemble architecture.
In this problem we have just 8 classes to predict, so another possibility could be to develop 8 different convolutional networks. Each of them trained to identify a specific class, and then, combine them together in a unique model able to classify the whole test set correctly. The single architecture should be specific and not too deep, otherwise, the ensemble model would explode in terms of size. Also, it is possible for a classification problem with a small number of labels, otherwise developing many different networks could become infeasible in practice.

References

[0] Perez, L., & Wang, J. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning. arXiv. link
[1] Mads Dyrmann, Henrik Karstoft, Henrik Skov Midtiby, Plant species classification using deep convolutional neural network, Biosystems Engineering, Volume 151, 2016, Pages 72-80, ISSN 1537-5110
[2] Cao, Xudong. "A practical theory for designing very deep convolutional neural networks." Unpublished Technical Report (2015)
[3] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. link
[4] Chollet, F. (2016). Xception: Deep Learning with Depthwise Separable Convolutions. arXiv. link
[5] Seungmin Han, Jongpil Jeong: An Weighted CNN Ensemble Model with Small Amount of Data for Bearing Fault Diagnosis, Procedia Computer Science, Volume 175, 2020, Pages 88-95, ISSN 1877-0509,
https://doi.org/10.1016/j.procs.2020.07.015. link