A Case Study on Overfitting in Multiclass Classifiers Using Convolutional Neural Networks

Yanexis P. Toledo[1], Thais Luca M. de Almeida[2], Flavia Cristina Bernardini[1], Eduardo de O. Andrade[1]

[1] Instituto de Computação
Universidade Federal Fluminense (UFF) – Niterói, RJ – Brazil

[2]Programa de Engenharia de Sistemas e Computação - COPPE
Universidade Federal do Rio de Janeiro (UFRJ) – Rio de Janeiro, RJ – Brazil

{ypupo, fcbernardini, eandrade}@ic.uff.br[1], tluca@cos.ufrj.br[2]

Abstract. *Convolutional Neural Networks (CNNs) have achieved much success mainly in areas of computational vision, such as image recognition, classification, object segmentation, and more. The learning process of this type of network generally requires large volumes of data, commonly high-resolution images, and the adjustment of a large number of parameters. The lack of control over the learning process of the model can lead to various problems. One of them is overfitting, which leads the network to a situation where it loses generality, making incorrect forecasts in the presence of new data. Another very common problem is its speed of convergence, which depends on the parameterization of the network: selection of the number of filters per layer, number of convolution layers, and more, where a fine adjustment is very important to avoid excessive computational costs. Understanding the origins of these problems and the ways to prevent them from happening is essential for a successful design. In this paper, we analyze these problems by designing a multiclass classifier among ten categories of images from the Caltech 256 dataset, based on the metrics of accuracy, precision, recall, and loss. To do so, Python 3.6, TensorFlow and Keras libraries were used on an RTX 2060 GPU.*

1. Introduction

In recent years, Convolutional Neural Networks (CNNs) have been achieving successful results in various areas of knowledge [Gong and Zhang 2016] [Yue 2017] [Luo et al. 2018] [Zolfaghari et al. 2018], mainly in computational vision area with patterns detection for classification, localization and image objects segmentation [Ge et al. 2018][Liu et al. 2018]. CNNs have become one of the state-of-the-art methods for image classification in various domains, especially on large datasets such as ImageNet. The success in creating a classification model, as well as its rapid convergence, depends on a good architecture design, which in turn depends on the dataset used during the training and the required output function.

CNN architectures for classification consist of two different functional architectural parts, which we call functional blocks. The first block is responsible for the image feature extraction and the second for classification. In this paper, we explore the design of these two functional blocks. The first by creating convolutional layers from

scratch, adjusting the number of layers (network depth), the number of filters per layer, and other aspects. We also use transfer learning to optimize robust architectures in feature extraction, allowing rapid convergence and performance improvements. As indicated in [Menegola et al. 2017] and [Shin et al. 2016], transfer learning has shown good results and is a more advantageous alternative, since it renounces the time it would devote to train a robust neural network from scratch for a specific domain. [Luo et al. 2018] explore the state of the art in the task of image classification, which consists of the use of architectures based on deep learning for the realization of transfer learning. The second functional block was explored by designing Multi-Layer Perceptrons (MLP) and Support Vector Machines (SVM) models. Nevertheless, CNNs may suffer overfitting during their learning process [Elleuch et al. 2016] [Xie et al. 2016], and how to avoid this problem is a constantly research issue.

Many regularization techniques have been developed to prevent neural networks from overfitting, e.g., L2 regularization [Krizhevsky et al. 2009], Dropout [Hinton et al. 2012], which discards randomly-selected activation's on the layers during training, DropConnect [Wan et al. 2013], which sets randomly-selected weights to zero during training, data augmentation, which manipulates the input data [Cireşan et al. 2010][Krizhevsky et al. 2012a], early stopping [Plaut et al. 1986], and DisturbLabel, which imposes the regularization within the loss layer [Xie et al. 2016]. In many cases, the use of a single technique is not enough to avoid network overfitting, so it is important to know how to recognize when networks start to overfit as well as infer the possible solutions.

In this paper, we present a case study of the overfitting process in multi-class classifiers. We develop several different architectures, identifying when they overfit as well as the possible solutions in each case. To do so, Caltech 256 dataset was used, yet we selected only the ten classes with most instances. The remainder of this paper is organized as follows. Firstly, Section 2 presents materials and methods used, the dataset description, background concepts and some existing CNNs architectures. Next, Section 3 presents the experiments performed and a brief discussion on the presented methods based on their quantitative results on the mentioned dataset. At last, Section 4 draws conclusions about this work and presents our future research directions.

2. Materials and Methods

This section describes the main characteristics of the architectures used in the design of the multi-class image classification model for Caltech 256 dataset.

2.1. Dataset Description

We evaluated our proposed network structures on dataset Caltech 256 [Griffin et al. 2007]. It is a set of 30607 images divided into 256 categories of objects that can be used in the development of different approaches in image recognition tasks [Puthenputhussery et al. 2017][Bodesheim et al. 2015][Banerji et al. 2013]. Caltech 256 has natural and artificial objects in various environments. Images can be found in different lighting conditions, positions, contexts, and sizes. 3398 (75% of the total) images were used for training and 1127 (25% of total) for testing. In this work, the ten categories with the highest number of images were chosen. All images were resized,

since they have original different dimensions and some architectures, used in this work, require specific dimensions.

2.2. Data Augmentation

Deep learning methods need a large amount of data during training. Then, to increase the dataset used, we also use data augmentation [Van Dyk and Meng 2001]. Data augmentation consists of applying image processing operations on each image of the dataset, generating five, ten, or more, new images from existing training data. In this work, all images were normalized and new images were generated by staggering, zooming and reversing the training images.

2.3. Convolutional Neural Networks

CNNs differ from classical perceptrons formulation by combining three architectural ideas to ensure some degree of shift, scale and distortion invariance: local receptive fields, shared weights and spatial sub-sampling [LeCun et al. 1999]. The main functional modules of any CNN are separated by layers, which are: an input layer, convolutional layer, pooling layer and a regular multilayer neural network called fully connected layer.

The input is an image with dimension [width x height x depth]. The depth corresponds to the color channels of the images: in the presence of a colored image, it consists of three channels (RGB), while in a grayscale image the depth is only one.

The convolutional layer is a linear model used to extract the input signal pattern. Its purpose is to generate feature maps using linear convolutional filters through a nonlinear activation function. Nonlinear Rectified Linear Units (ReLU) activation function is often used in the hidden layers, while SoftMax activation function is used in the final layer. By stacking layers of linear and nonlinear functions, we can detect a wide range of patterns and accurately predict a label for a given image. SoftMax is often used to produce a discrete probability distribution vector and the output corresponds to the probability that a given image corresponds to a particular class.

After the convolution layers, pooling layers are used to reduce the dimensions of feature maps and, consequently, reduce computation cost and avoid overfitting, besides creating invariance to small changes and local distortions. There are several forms of pooling applicable to a feature map: max pooling (selects a maximum value), average pooling (takes the average), and more. At the end, we have a fully connected layer, which is responsible for tracing the decision path between classes, followed by classification functions that influence the learning of filters and, consequently, the network results.

CNNs may also contain the so-called local contrast normalization layers (LCN), which are positioned at the output of the pooling layers and normalize the contrast of an image in a non-linear way. Instead of a global normalization (i.e., considering the whole image), LCN applies normalization over local image regions, considering each pixel at a time. Normalization may correspond to subtracting the neighborhood mean from a particular pixel and dividing by the variance of the pixels values of that neighborhood. This transformation equips CNNs with the invariance of brightness, useful property in the context of object recognition in images [Jarrett et al. 2009].

2.3.1. VGG

VGG is a deep CNN model, developed by [Simonyan and Zisserman 2014], to model the ImageNet 2014 for the Large Scale Visual Recognition Challenge (ILSVRC-2014). VGG consists of sixteen convolutional layers and impresses by its uniform architecture. It is similar to AlexNet [Krizhevsky et al. 2012b], but with 3x3 convolutions. AlexNet is a deep CNN model that consists of five convolutional layers, where ReLU layer is applied after each convolutional layer. The first, second and fifth layers contain max-pooling layers.

2.3.2. Inception-V2

Inception-V2 is a CNN model, developed by [Szegedy et al. 2016], to model ImageNet 2012 for ILSVRC-2012. Inception network was an important milestone in the development of CNN classifiers. Prior to its inception (pun intended), most popular CNNs just stacked convolutional layers deeper and deeper, hoping to get better performance. The Inception network, on the other hand, was complex (heavily engineered). It used a lot of tricks to push performance; both in terms of speed and accuracy. Its constant evolution lead to the creation of several versions of the network. The Inception networks come to solve the difficulty of selecting kernels size of convolution layers, allowing to use several sizes at the same time, which facilitates the features detection, independently of the objects dimensions in the images.

2.3.3. ResNet

ResNet is a CNN model developed by [He et al. 2016], to model ImageNet 2015 for ILSVRC 2015. Instead of learning a direct mapping $x \rightarrow y$ with a function H(x), it defines a residual function $F(x) = H(x) - x$, which may be refracted in $H(x) = F(x) + x$, where F(x) e x represent the stacked nonlinear layers. ResNet models are implemented with single layer jumps. Jump layers avoid the vanishing gradient problem, reusing activations from an earlier layer until the adjacent layer learns its weights. The use of few layers simplifies network training in the initial stages, that speeds up learning by reducing the impact of vanishing gradients as there are fewer layers to propagate. The network gradually restores the skipped layers as you learn the resource space.

2.4. Transfer Learning

Learning a new model from scratch requires a huge amount of data and high processing power. Therefore, pre-trained networks are used as the starting point on many tasks and can also be used as feature extractors by just removing its fully-connected layer, which is responsible for learning specific details of the dataset with which the network was trained.

Transfer learning for image classification tasks consists of using the synaptic weights configured by a network to recognize and extract the characteristics of a different dataset from the one that it was previously trained. This alternative is very attractive, mainly when the dataset to be classified has similar characteristics to another dataset used to train a pre-trained network and when we have to deal with a training dataset that is not

big enough. In addition to this, a robust feature extraction is achieved without the need to waste time in parameterizing a network from scratch [Yosinski et al. 2014].

3. Experiments and Discussion

In order to assess the performance and robustness of CNNs, we conduct experiments on the dataset described in Section 2.1 to study the performance of different architectures. Then, to check effectiveness of fine-tuned networks, we use VGG-16, VGG-19, Inception V2 e ResNet 50 models, fine-tuned using ImageNet dataset [Deng et al. 2009]. For our experiments, we used Keras and TensorFlow in Python 3.6 and RTX 2060 GPU.

3.1. Increasing Network Depth

Initially, we have created a CNN consisting of three convolution layers followed by max-pooling layers of 2x2 dimension. Each convolution litter consists of 64 filters of 3x3 dimension. At the end, was used a fully connected layer with 500 neurons. In the training process, all weights were randomly initialized, Adam optimizer algorithm and learning rate of 0.0005 were used. This first network presents an accuracy of 76%, and as seen in Figures 1a and 1b, it's susceptible to overfitting problem. The class distribution for training and test datasets, can be seen in Table 1.

Table 1. All the instances by class for training and test datasets.

Class	Training	Test
Clutter	621	206
Airplane	800	200
Motorcycle	798	199
Face	435	108
T-Shirt	358	89
Hammock	285	71
Snooker	278	69
Horse	270	67
Ladder	242	60
Bathtub	232	58

Increasing the dataset through data-augmentation and adding a Dropout layer of 50% at the end of the fully connected layer has boosted the accuracy by 4% during tests. It's still susceptible to overfitting after epoch 10. We could use early stopping before epoch 10 and save the current model, avoiding the final model to get to epoch 20, when it loses generality. Instead, we decided to explore the parametrization of the network in order to get greater precision and accuracy. Our experiments have shown that increasing the number of filters per layer doesn't provide significantly superior results and also slows down the learning process. For that reason, another strategy used to avoid overfitting was to decrease the number of neurons in the MLP output by five times, which eliminates the problem during the 20 training epochs, as shown in Figures 1c and 1d. As consequence, we obtain a bit lower accuracy of 78% during tests.

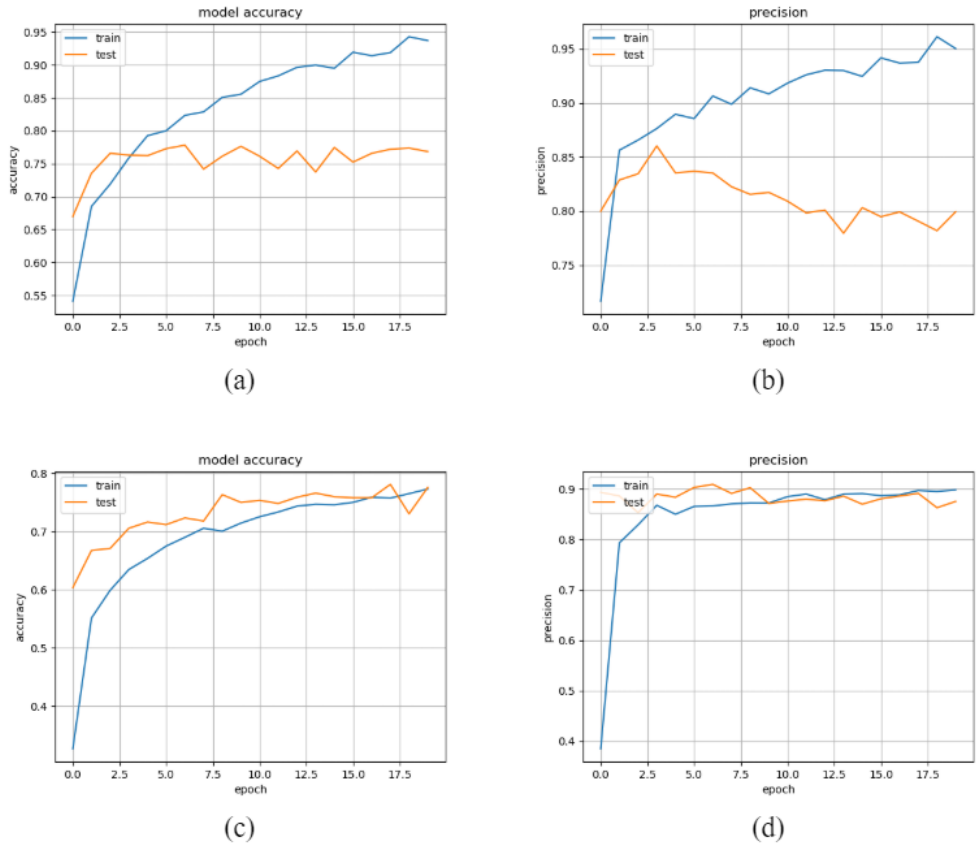

Figure 1. (a) and (b) show results using a fully connected layer with 500 neurons. For accuracy and precision results, it's susceptible to overfitting problems. When the number of neurons in the MLP output layer is decreased by five times, the overfitting problem is eliminated, as shown in (c) and (d).

3.2. Using Transfer Learning

In order to study how CNNs are susceptible to overfitting, we used VGG-16, VGG-19, Inception-V2 and ResNet 50 models pre-trained using ImageNet dataset, which consists of 1000 different classes. We choose VGG-16 because it is the inspiration for the next versions of existing architectures. VGG-19 has the same concept of VGG-16, except for its depth of 19 layers, and was used to analyse how depth influences in the results. Inception-V2 and ResNet 50 have been attractive for introducing a new concept on CNNs.

It is in the fully connected layer that specific information about the dataset is learned, so we decided to replace their outputs by a MLP dense layer of 100 neurons and used SoftMax activation function to calculate the ten probabilities of a image belongs to of the 10 categories. The first layers work as generic extractors which can be used for different tasks. It was possible to use pre-trained networks because ImageNet is similar to Caltech 256. Training from scratch would be very computationally expensive, so we trained only the aggregate layers in the output classification.

The results obtained for VGG-16 are very similar to those obtained for VGG-19, maybe because the only difference between the two is their depths. VGG-16 was a bit faster than VGG-19 for 50 epochs, for example. VGG-16 reaches an accuracy of 85% while VGG-19 has an accuracy of 83% with the same processing time. Inception-V2 and

ResNet 50 are faster and have more precision.

All networks had achieve high levels of precision, accuracy and low levels of error, but ResNet 50 has the finer and robustest behavior, as shown in Figure 2, and reaches the highest level of precision with faster convergence and shorter processing time. It performs 20 epochs in 17 minutes on GPU.

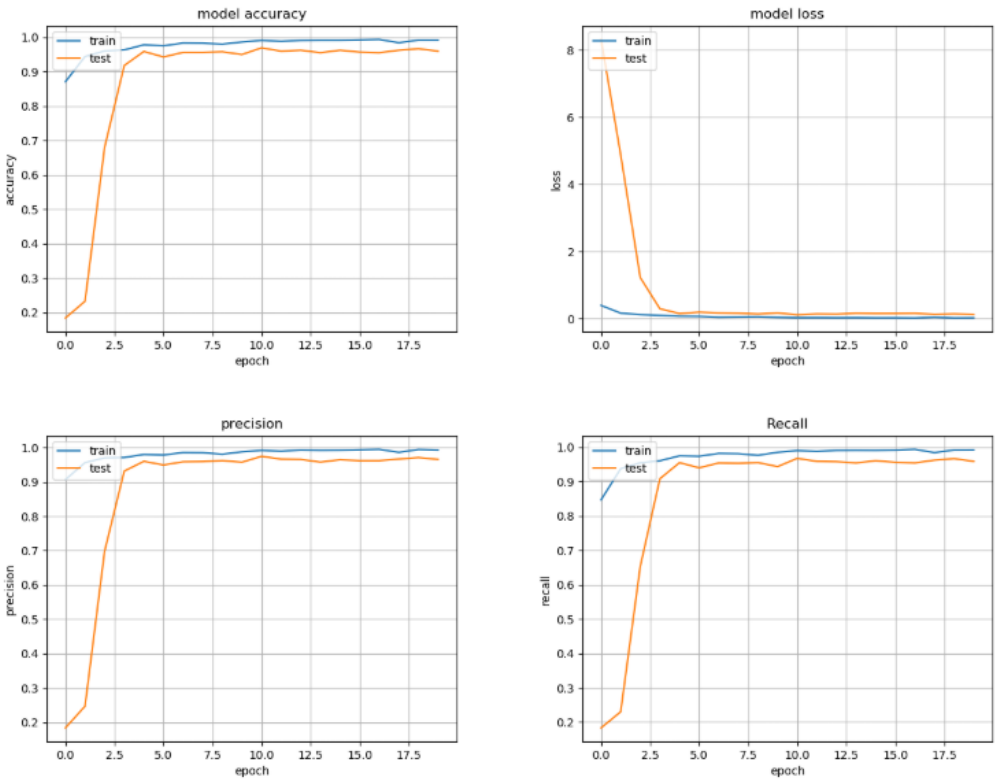

Figure 2. ResNet50 model accuracy, precision, loss, and recall using transfer learning and SoftMax output function.

3.3. Using SVM in the Output Layer

As CNNs are composed of two basic parts, feature extraction and classification, we decided to explore CNNs as features extractors and use a SVM linear model as classifier. We also wanted to evaluate the performance of SVM working with a more complex, robust and good performance extractor.

The results using ResNet50 remained great when using SVM as classifier. Inception-V2 has a significant improvement in its results when SVM is used for classification. Figure 3 shows the results for ResNet50 and Figure 4 shows the results for Inception-V2. Analysing the graphs, we can see that the great advantage in using SVM, instead of a fully connected network, for classification is in time of training and execution of the tests. Table 2 compares the results of the best models trained throughout our study and also compares the time of training and testing for each model.

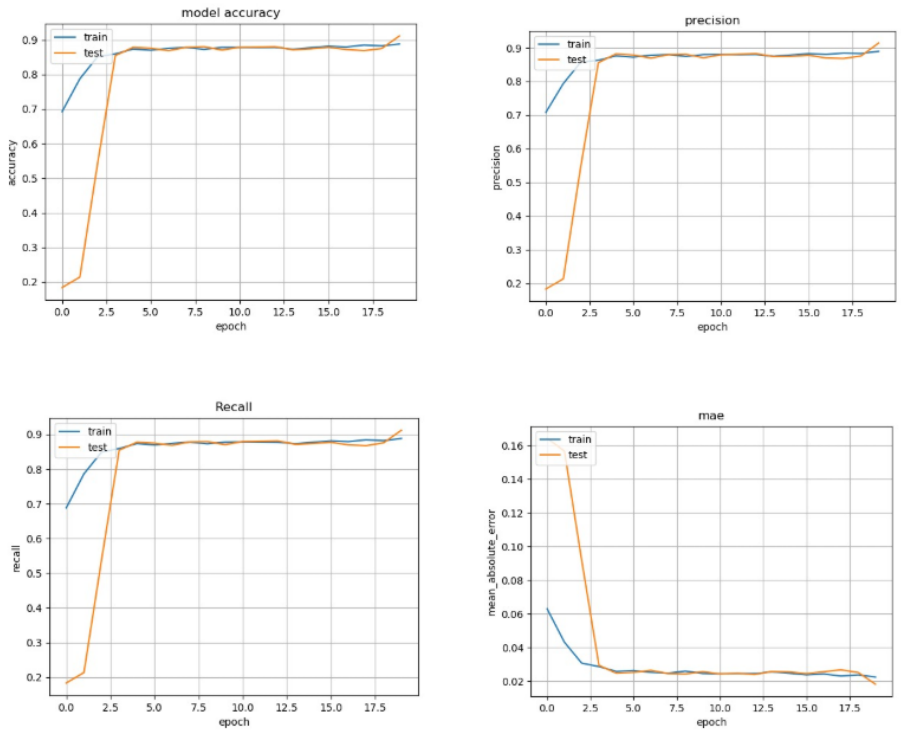

Figure 3. ResNet model accuracy, precision, recall, and MAE during training and test using SVM as its final layer.

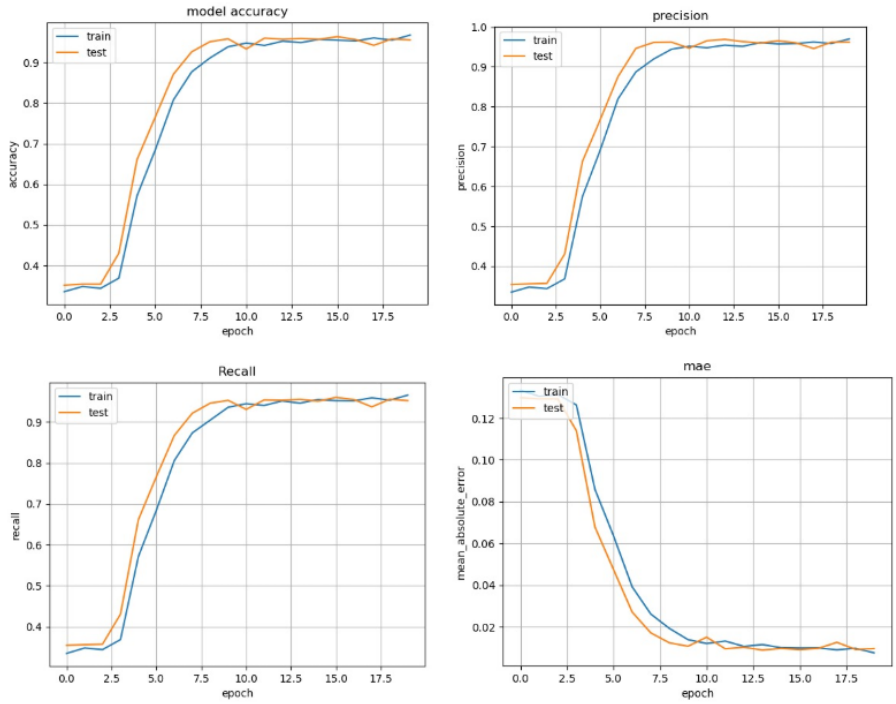

Figure 4. InceptionV2 model accuracy, precision, recall, and MAE during training and test using SVM as its final layer.

Table 2. Comparison of different architectures used by accuracy, precision, recall, loss, MAE and time consumed.

Architecture	Accuracy	Precision	Recall	Loss	MAE	Time Consumed
Basic CNN	81%	86%	79%	0,7	0,05	10 min
TL-VGG-16	85%	86%	77%	0,4	0,03	39 min
TL-VGG-19	83%	85%	76%	0,5	0,04	42 min
TL-InceptionV2	94%	95%	93%	0,19	0,015	33 min
ResNet 50	96%	97%	97%	0,2	0,01	17 min
ResNet50 + SVM	96%	96%	96%	0,9	0,01	16 min
TL-InceptionV2+SVM	96%	95%	96%	0,15	0,09	22 min

4. Conclusion and Future Work

When developing a Deep Neural Network model, we must take into account overfitting problem, and avoiding this problem is important so the network can correctly generalize based on the examples learned. In this paper, we evaluate how some multi-class classifier architectures behave when facing the problem and present different strategies that can be applied to handle overfitting.

We also show that the success of a multi-class classifier design depends on both the architecture responsible for extracting features and the output layer for classification. Some configurations of the two functional parts were evaluated obtaining the best results using ResNet for feature extraction and SVM for image classification.

As expected, the deepest network, in terms of convolutional layers, improves the results in image classification. This happens because as we increase the number of convolution layers, we manage to extract more complex characteristics of the data, which helps the output layers to distinguish better between classes, especially when we have large numbers of output classes. On the other hand, deeper networks are more computationally expensive.

In the future, we plan to use more tests to also analyze the performance of more recent CNNs: use the winners of the last edition of ImageNet Large Scale Visual Recognition Challenge (ILSVRC), like NUS Qihoo UIUC DPNs (VID), winner of classification and object localization tasks in ILSVRC 2017. We could also try to boost the performances of our models using a bigger dataset during training, since our majority class has only 827 images. Furthermore, the design of MLPs used in the last layer can also be explored, in this work, our MLP consisted of 256 neurons and then was reduced to 100 to avoid overfitting, so we believe it can reach a best fit.

References

Banerji, S., Sinha, A., and Liu, C. (2013). New image descriptors based on color, texture, shape, and wavelets for object and scene image classification. *Neurocomputing*, 117:173–185.

Bodesheim, P., Freytag, A., Rodner, E., and Denzler, J. (2015). Local novelty detection in multi-class recognition problems. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 813–820. IEEE.

Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, 22(12):3207–3220.

Deng, J., Dong, W., Socher, R., jia Li, L., Li, K., and Fei-fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *In CVPR*.

Elleuch, M., Maalej, R., and Kherallah, M. (2016). A new design based-svm of the cnn classifier architecture with dropout for offline arabic handwritten recognition. *Procedia Computer Science*, 80:1712–1723.

Ge, Y., Li, B., Zhao, Y., Guan, E., and Yan, W. (2018). Melanoma segmentation and classification in clinical images using deep learning. In *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, pages 252–256. ACM.

Gong, Y. and Zhang, Q. (2016). Hashtag recommendation using attention-based convolutional neural network. In *IJCAI*, pages 2782–2788.

Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 object category dataset. *CalTech Report*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Jarrett, K., Kavukcuoglu, K., LeCun, Y., et al. (2009). What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012a). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. (1999). Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer.

Liu, W., Liao, S., Hu, W., Liang, X., and Chen, X. (2018). Learning efficient single-stage pedestrian detectors by asymptotic localization fitting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 618–634.

Luo, C., Chu, X., and Yuille, A. (2018). Orinet: A fully convolutional network for 3d human pose estimation. *arXiv preprint arXiv:1811.04989*.

Menegola, A., Fornaciali, M., Pires, R., Bittencourt, F. V., Avila, S., and Valle, E. (2017). Knowledge transfer for melanoma screening with deep learning. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 297–300. IEEE.

Plaut, D. C. et al. (1986). Experiments on learning by back propagation.

Puthenputhussery, A., Liu, Q., and Liu, C. (2017). A sparse representation model using the complete marginal fisher analysis framework and its applications to visual recognition. *IEEE Transactions on Multimedia*, 19(8):1757–1770.

Shin, H.-C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., and Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Van Dyk, D. A. and Meng, X.-L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066.

Xie, L., Wang, J., Wei, Z., Wang, M., and Tian, Q. (2016). Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4753–4762.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Yue, S. (2017). Imbalanced malware images classification: a cnn based approach. *arXiv preprint arXiv:1708.08042*.

Zolfaghari, M., Singh, K., and Brox, T. (2018). Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 695–712.