ISSN 1870-4069

Comparative Study of Optimizers in the Training of a Convolutional Neural Network in a Binary Recognition Model

Marco López-Sánchez, José Hernández-Torruco, Betania Hernández-Ocaña, Oscar Chávez-Bosquez

Universidad Juárez Autónoma de Tabasco,
División Académica de Ciencias y Tecnologías de la Información,
Cunduacán, Tabasco,
Mexico

{marco.lopezsanchez,oscar.chavez,jose.hernandezt,
betania.hernandez}@ujat.mx

Abstract. Comparative study of optimizers in the training of a convolutional neural network in a binary recognition model. In Machine and Deep learning, the optimizer selection is a crucial step, as the model performance depends heavily on the optimizer selected. Deep neural network architectures often have multiple layers of representations used to learn from training data automatically. Hyperparameter calibration plays an essential role in the learning process. As the parameter values may vary depending on the data set, we want to obtain an essential improvement in the model performance in the training phase. This work's main objective was to determine which optimizer can obtain the best accuracy in the training phase of a convolutional neural network used to perform binary classification using the public dataset Dogs vs. Cats, consisting of a total of 25,000 images. The compared optimizers are SGD, Adam, and RMSprop. The model architecture developed consists of three convolution layers with their corresponding maxpooling layer. As a result, experiments show that the Adam optimizer offers better performance. These results suggest that the use of Adam to train binary convolutional neural networks should be considered.

Keywords: Image classification, optimizer, deep learning, convolutional neural networks.

1 Introduction

The implementation of computer vision has attracted the interest of researchers within the area of computer science, as Deep learning has become a promising area in the field of computer vision [10]. Deep learning uses Artificial neural networks internally as the main algorithm to perform predictions, classification, and so on. Image classification, covering a diverse range of applications including biometrics [12], video surveillance [18] and medical research [14].

Among the most promising algorithms, the convolutional neural network (CNN) technique [15], is the most widely used deep learning algorithm. In this paper, we describe a binary classification problem and analyze the Adam, RMSprop, and SGD optimizers, which are used to train deep learning neural networks. ADAM is the most recent and represents an improvement of other optimizers, but it is not always the one that generates the best model. Following, we present key concepts and optimization techniques, followed by a brief description of neural convolution networks. The rest of the work is organized in the following order: Section 2 materials and methods in section 2. The results of the experiments are presented in section 3. Finally, section 4 concludes the article.

1.1 Artificial Neural Networks

Artificial neural networks (ANN) aim to perform computational tasks using a large number of simple interconnected processing units called neurons or nodes. The connections between neurons are associated with parameters called weights that can be modified, through a training process, to associate the desired output to a specific input [4]. It can be described as a directed graph in which each node performs a transfer function of the form [23].

Hyperparameters are a particular set of parameters used in the learning process during training. Hyperparameters are vital since they directly impact the neural network's training phase and hence in the model performance. There are several hyperparameter optimization techniques, but it is the data scientist's responsibility to initialize these values manually. The parameters might be an integer or a continuous or categorical variable which ranges from the lower to upper bound values [22].

The basic hyperparameters of an ANN are:

- Number of hidden layers: Neural networks with single-layer tunable parameters are very limited in what they can do, as is the case with perceptrons. Therefore, it is natural to expand a neural network's capacity by adding additional layers of neurons. From outside the network, only the first and last layers of a multi-layer network are visible: the input and output layers. All other layers are "hidden" layers". The additional layers are therefore called hidden layers because they are not visible from the outside [2].
- Number of hidden units: Each hidden layers can have a different number of neurons.
- Learning rate: The learning rate is a small number, often between 0.00001 and 0.05, which affects the magnitude that is added to the current weight of an edge in order to train the model with these updated weights [5]. In nearly all gradient descent algorithms, the choice of learning rate remains central to efficiency. Yoshua Bengio [1] claims that it is often the single most crucial hyper-parameter and that it always should be tuned.

- Number of epochs: One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches. For example, an epoch that has one batch is called the batch gradient descent learning algorithm [3].
- Batch size: The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters [3].
- Loss function: This is a tool to measure a decision algorithm's performance (or classifier). The loss function measures each classifier action's cost and converts an error probability error into a decision. This approach allows us to handle situations where particular classification mistakes have to be considered differently from the others [4].
- Activation function: This is critical as images and features within an image are highly non-linear problems, and most other functions within Convolutional neural networks (Conv2D, pooling, fully connected layers, and so on) generate only linear transformations. The activation function generates the non-linearity while mapping input values to its ranges. Many activation functions are used, but the most common ones are Sigmoid, Tanh, ReLU.

An optimizer's role is to update the weight parameters then to minimize the error function or loss function, where the error is the difference between the actual value and the predicted value. Before starting the training phase, the optimizer that carries out this task must be selected along with its specific algorithmic hyperparameters. The optimizers compared in this study are:

SGD The Stochastic Gradient Descent algorithm [19] is one of the earliest methods used for training neural networks. The main advantage of these methods is the simplicity of each iteration, both in generating the search direction and in performing the update of variables [17].

RMSprop It was introduced to address the monotonically decreasing learning rate problem. This problem is present in the AdaGrad optimizer [11]; it uses exponential decay in the first step.

Adam The Adam optimization [13] algorithm was introduced to combine the benefits of AdaGrad, and RMSProp algorithms.

The performance metrics used in this work are:

- Accuracy: It is the most used metric to summarize the performance of a supervised learning model, as it indicates the proportion of examples that a classifier can classify correctly, usually indicated as a percentage [2].
- Confusion matrix: In its simplest form, a confusion matrix (also called an error matrix) is a type of contingency table with two rows and two columns that contains the numbers of false positives, false negatives, true positives, and true negatives [5].
- ROC curve: The ROC (receiver operating characteristic) curve is a curve that plots the TPR (true positive rate, i.e., the recall) against the FPR (false positive rate) [5].

1.2 Convolutional Neural Networks

A convolutional neural network (CNN) is a deep learning algorithm specialized in image classification. The core element of CNNs is the processing of data through the convolution operation. Convolution of any signal that is combined with another one produces a third signal that may reveal more information about the signal than the original signal itself [6].

In 1990, LeCun et al. [16] published the seminal paper fundamenting the modern framework of a CNN, and later improved it in [15]. The specific hyper-parameters of CNN are [22]:

– Number of convolutional layers. This layer is the primary building block of a convolutional neural network, which determines the output of associated.
– Number of pooling layers. The primary function of the Pool layer is to progressively reduce the spatial size (i.e., width and height) of the input volume. Typically, we use a pool size of 2×2, although deeper CNNs that use larger input images (> 200 pixels) may use a 3×3 pool size early in the network architecture.
– Step size handled by the stride. It is the number of pixels to move to define the local receptive field for a filter–a stride of 1 means to move across and down a single pixel. The value of stride should not be too small or too large, and it is almost always symmetric in the dimensions of height and width.

A convolutional neural network, also known as ConvNet or CNN, is a variant of the artificial neural network, which specializes in emulating functionality and behavior of our visual cortex [21]. CNN has produced excellent results in fields like image classification [15], speech recognition [9] and object identification [7].

Typical CNN architectures stack a few convolutional layers (each one generally followed by a ReLU layer), then a pooling layer, then another few convolutional layers (+ReLU), then another pooling layer, and so on [8].

When CNNs are applied to images, consecutive convolutional layers learn progressively abstract features. Figure 1 depicts this process, where the first layers take care of extracting characteristics such as the edges or corners of an image. Once these edges are detected, they are used to detect shapes in the subsequent layers. When the shapes are detected, high-level characteristics are detected, such as a wheel in an image where a car appears. The last layers are fully connected and take care of giving a prediction from these last extracted characteristics.

2 Materials and methods

2.1 Dataset

The Dogs vs. Cats dataset was originally collected by Microsoft and consists of more than 3 million images, of which 25,000 have been publicly released[1]. It is

[1] https://www.microsoft.com/en-us/download/confirmation.aspx?id=54765

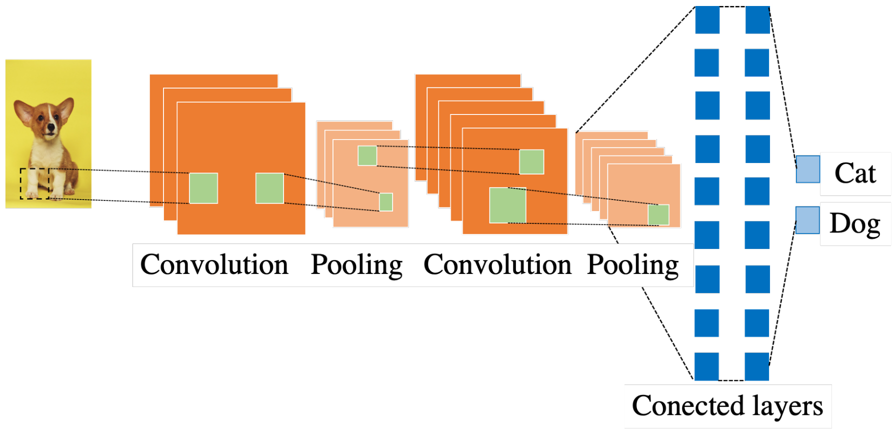

Fig. 1. Example of a convolutional neural network.

worth mentioning that there is a smaller dataset (3,000 images) derived from this one, that is more commonly used in the literature.

Data are the images, which are stored in 2 directories, one for the pictures of cats and the other one for pictures of dogs. The dataset is balanced, i.e., it has the same number of images of cats and dogs. The goal is to build a machine learning algorithm capable of correctly detecting the animal (dog or cat) in the images.

2.2 Experimental setup

The CNN model proposed in this paper is used to predict the correct class of each item in the dataset. The data were split into 70 % training, 15 % for validation, and 15 % for testing. The total number of convolutional layers is 3*stacks. Each stack has a convolutional layer, a batch normalization layer, and a ReLU layer. The image size is $150*150*3$, where $150*150$ is the size in pixels, and the number 3 represents the depth of the image.

The hyperparameter configuration was:

- Learning rate: 0.00001,
- Number of epochs: 30,
- Steps per epoch: 70,
- Loss function: Binary crossentropy,
- Activation functions: ReLU and Sigmoid.

The life cycle of a model provides the backbone for modeling a dataset. The life cycle used in this work includes [20]:

1. Gather the dataset: download a zip file containing the dataset, unzip it and discard the corrupt files.

2. Split the dataset: divide the data set into three parts:
   - Training set: the set of samples used to train the model.
   - Validating set: the set of samples that we are going to use to adjust the parameters of the classifier.
   - Testing set: the set of samples that we will use only to evaluate the performance of the classifier.

To the training set we assign 70 % of the images, to the validating 15 % and to the testing 15 %. Train network: In this step, we define the architecture of our network configuring each of its layers. We also configure the hyperparameters, then compile the model and feed it with the training set. Evaluate: To verify the result obtained in training, we use the confusion matrix and the ROC curve. Make predictions: We provide a simple user interface so that the user can load an image and test the prediction in a more intuitive way.

3 Results

We conducted experiments with the dataset to determine the behavior of each optimizer. To test the performance of each optimizer in all the experiments, the default settings of each hyperparameter were chosen. We have included an early stopping in the training phase to stop the training loop after 5 iterations with no improvement. Table 1 shows the main performance metrics on each optimizer. Table 1 shows that the highest accuracy was achieved by the Adam optimizer, while RMSprop was second with a small gap. SGD fall behind with a weak accuracy.

Table 1. Model results.

Optimizer	**Loss**	**Accuracy**
SGD	0.6926	0.5053
RMSprop	0.4520	0.7868
Adam	0.4227	0.8017

Figure 2 shows the performance of each model in the training phase on the training and validation subsets. We highlight that SGD stops training at epoch 12, according to the early stopping configuration. On the other side, RMSprop and Adam continue the training until the 30 configured epochs. SGD validation curve shows an erratic behavior, while RMSprop and Adam curves shows the slope of the learning process.

The confusion matrix of each optimizer is presented in Figure 3. We can notice that SGD has the lower performance, misclassifying most of the images. On the other hand, RMSprop and Adam performed quite well in the classification of cats and dogs.

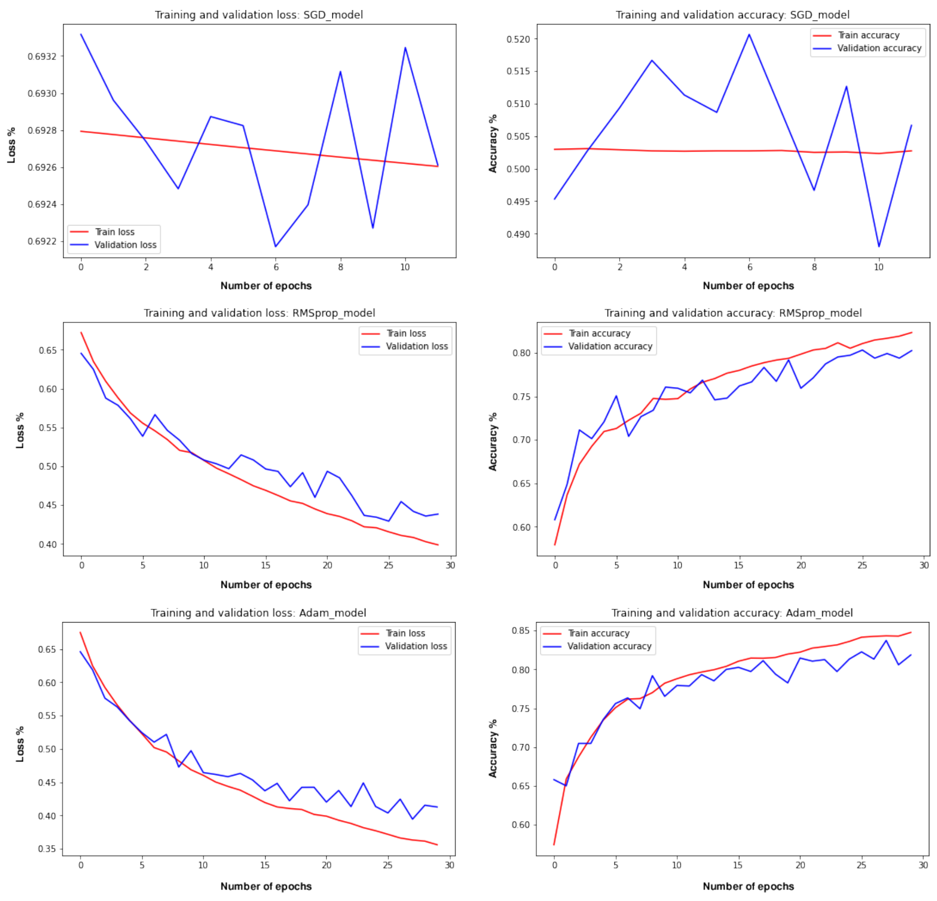

Fig. 2. Optimizers performance.

Figure 4 shows the graphical representation with normalized values of the confusion matrix of each optimizer. In these plots the difference between the correct classifications (top left and bottom right of each plot) against misclassifications (bottom left and top right) is clear. For example, we can see that SGD virtually classifies all the images as dogs, having a high rate with dog images but a low rate classifying cat images.

In Figure 5 we can see the area under the curve (AUC) of the ROC curve of each optimizer, computing the false-positive rate and the true-positive rate of each model. The AUC measures how much the model is capable of distinguishing between cats and dogs. We can notice the flattened curve in the SGD optimizer, indicating a poor classification performance. The ROC curve for RMSprop and Adam is similar, along with their AUC. Both optimizers performed well in the classification task.

		SGD optimizer			RMSprop optimizer			Adam optimizer	
		Predicted value			Predicted value			Predicted value	
		Cat	Dog		Cat	Dog		Cat	Dog
Real value	Cat	38	1838	Cat	1392	484	Cat	1624	252
	Dog	18	1858	Dog	316	1560	Dog	492	1384

Fig. 3. Confusion matrix of each optimizer.

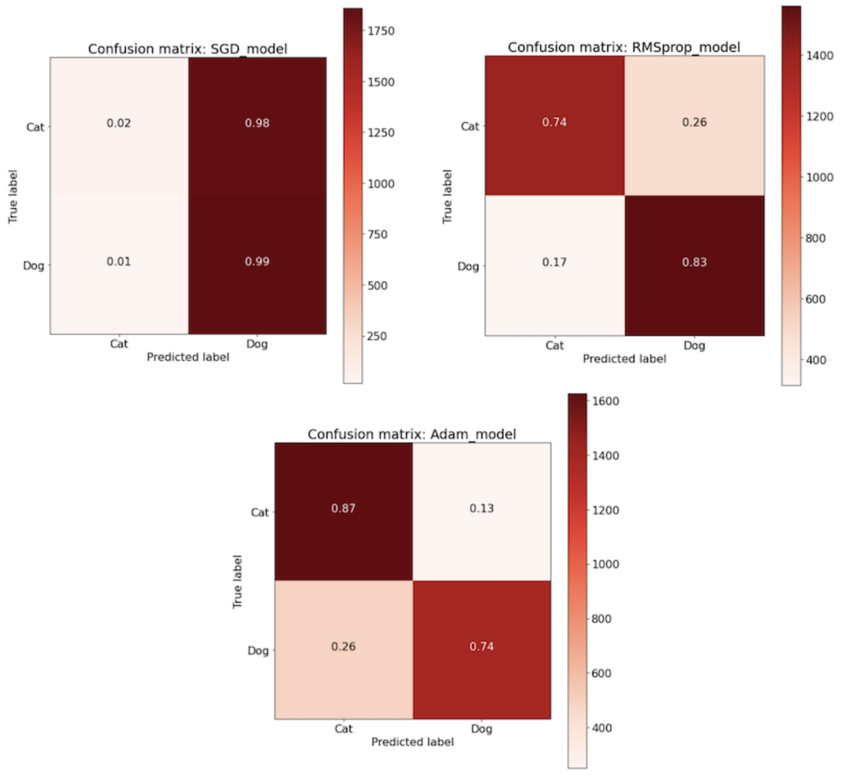

Fig. 4. Graphical representation of the confusion matrix.

4 Conclusions

Selecting an optimizer for training a neural network is a challenging task. The role of an optimizer is to update the weight parameters so the network can minimize the error or loss function, where the error is the difference of actual value and the predicted value.

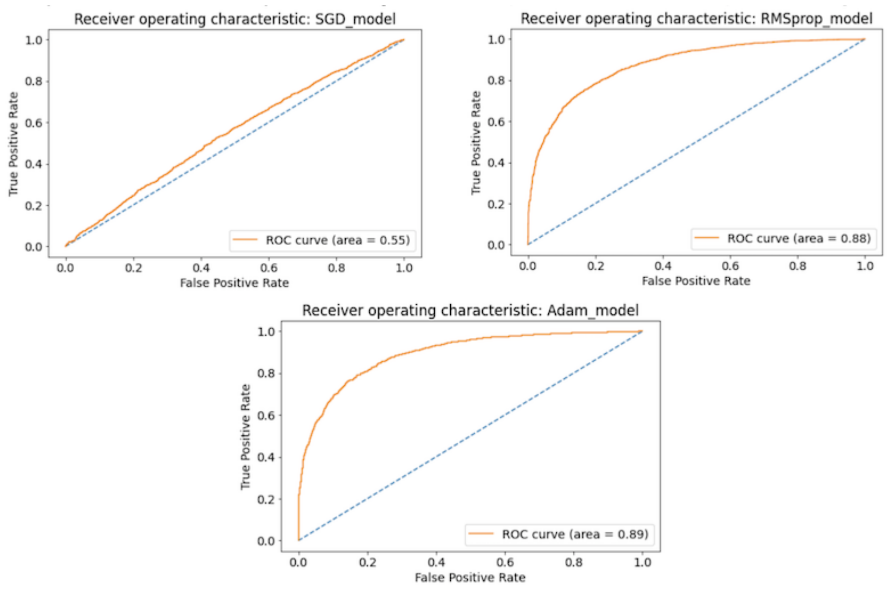

Fig. 5. ROC curve of each optimizer.

In this work, we test three optimizers: SGD, RMSprop, and Adam, creating a model for binary classification in the Dogs vs. Cats dataset. In our experiments, the Adam optimizer obtained the best performance, according to the model's accuracy and the AUC of the ROC curve. This may be due to the fact that Adam is an adaptative learning rate method, as it computes individual learning rates for diferent parameters. Adam has also the best performance in terms of speed of training.

Each optimizer has several parameters that have a direct impact on the performance of the generated model. In a future work, we consider the tuning of these parameters and using more and diverse optimizers.

Acknowledgments. To the Consejo Nacional de Ciencia y Tecnología (CONACYT) for supporting the Doctoral program in Computer Science at the Universidad Juárez Autónoma de Tabasco.

References

1. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In: Neural networks: Tricks of the trade, pp. 437–478. Springer (2012)
2. Berzal, F.: Redes neuronales & Deep Learning: Volumen II. Edición Independiente, 1era. edn. (2019)
3. Brownlee, J.: Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch. Jason Brownlee, 1st edn. (2016), https://books.google.com.mx/books?id=PCJnAQAACAAJ

4. Camastra, F., Vinciarelli, A.: Machine learning for audio, image and video analysis: theory and applications. Springer (2015)
5. Campesato, O.: Artificial Intelligence, Machine Learning, and Deep Learning. Stylus Publishing, LLC (2020)
6. El-Amir, H., Hamdy, M.: Deep Learning Pipeline: Building a Deep Learning Model with TensorFlow. Apress (2019)
7. Erhan, D., Szegedy, C., Toshev, A., Anguelov, D.: Scalable object detection using deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2147–2154 (2014)
8. Géron, A.: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media (2019)
9. Guiming, D., Xia, W., Guangyan, W., Yan, Z., Dan, L.: Speech recognition based on convolutional neural networks. In: 2016 IEEE International Conference on Signal and Image Processing (ICSIP). pp. 708–711. IEEE (2016)
10. Hinton, G., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation 18(7), 1527–1554 (2006)
11. Hinton, G., Srivastava, N., Swersky, K.: Neural networks for machine learning lecture: Lecture 6a overview of mini-batch gradient descent. Coursera (2012), online
12. Jaseena, K., Kovoor, B.: A survey on deep learning techniques for big data in biometrics. International Journal of Advanced Research in Computer Science 9(1), 12–17 (2018)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
14. Lakhani, P., Sundaram, B.: Deep learning at chest radiography: automated classification of pulmonary tuberculosis by using convolutional neural networks. Radiology 284(2), 574–582 (2017)
15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521, 436–444 (2015)
16. LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., Jackel, L.D.: Handwritten digit recognition with a back-propagation network. In: Advances in neural information processing systems. pp. 396–404 (1990)
17. Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM Journal on Optimization 22(2), 341–362 (2012)
18. Ojha, S., Sakhare, S.: Image processing techniques for object tracking in video surveillance-a survey. In: 2015 International Conference on Pervasive Computing (ICPC). pp. 1–6. IEEE (2015)
19. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics pp. 400–407 (1951)
20. Rosebrock, A.: Deep Learning for Computer Vision with Python: Starter Bundle. PyImageSearch (2017)
21. Sarkar, D., Bali, R., Sharma, T.: Practical machine learning with python. A problem-solvers guide to building real-world intelligent systems. Apress, Berkely (2018)
22. Victoria, A.H., Maragatham, G.: Automatic tuning of hyperparameters using bayesian optimization. Evolving Systems (2020)
23. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE 87(9), 1423–1447 (1999)