International Journal of Electrical and Computer Engineering (IJECE)
Vol. 12, No. 6, December 2022, pp. 6363∼6372
ISSN: 2088-8708, DOI: 10.11591/ijece.v12i6.pp6363-6372

Comparative study of optimization algorithms on convolutional network for autonomous driving

Fernando Martínez, Holman Montiel, Fredy Martínez
Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Bogotá D.C, Colombia

Article Info

Article history:

Received Oct 6, 2021
Revised Jun 4, 2022
Accepted Jul 1, 2022

Keywords:

Autonomous driving
Function optimization
Intelligent sensor
Path planning
Recall

ABSTRACT

The last 10 years have been the decade of autonomous vehicles. Advances in intelligent sensors and control schemes have shown the possibility of real applications. Deep learning, and in particular convolutional networks have become a fundamental tool in the solution of problems related to environment identification, path planning, vehicle behavior, and motion control. In this paper, we perform a comparative study of the most used optimization strategies on the convolutional architecture residual neural network (ResNet) for an autonomous driving problem as a previous step to the development of an intelligent sensor. This sensor, part of our research in reactive systems for autonomous vehicles, aims to become a system for direct mapping of sensory information to control actions from real-time images of the environment. The optimization techniques analyzed include stochastic gradient descent (SGD), adaptive gradient (Adagrad), adaptive learning rate (Adadelta), root mean square propagation (RMSProp), Adamax, adaptive moment estimation (Adam), nesterov-accelerated adaptive moment estimation (Nadam), and follow the regularized leader (Ftrl). The training of the deep model is evaluated in terms of convergence, accuracy, recall, and F1-score metrics. Preliminary results show a better performance of the deep network when using the SGD function as an optimizer, while the Ftrl function presents the poorest performances.

Corresponding Author:

Fredy Martínez
Facultad Tecnológica, Universidad Distrital Francisco José de Caldas
Carrera 7 No 40B-53, Bogotá D.C., Colombia
Email: fhmartinezs@udistrital.edu.co

1. INTRODUCTION

Robotics, control, intelligent systems, and artificial vision systems have been developing at a dizzying pace in recent years thanks to the development of deep learning models [1]–[3] These developments have been integrated into the automotive industry, and in particular, have led to a breakthrough in autonomous driving systems [4]–[6]. This type of technology offers human benefits such as a reduction in traffic accidents and congestion in large cities, and therefore an improvement in the mobility and quality of life of citizens [7], [8]. Linked to these advances are also environmental benefits, as autonomous systems have better management of energy resources [9], [10]. There are five levels of autonomous driving defined according to the SAE J3016 standard [11], with the lowest levels considered basic driver assistance and the highest levels aligned with autonomous systems that do not require human intervention. An important feature of all these vehicles, regardless of their level, is the degree of development of their sensors and intelligent control capabilities [12]. Basic navigation schemes can be solved from classical motion control schemes supported by distance sensors [13]. However, an autonomous driving system is confronted with a larger number of much more complex situations in which the well-being of the human being must also be ensured [14]. These are systems that require complex real-time decision-making schemes, which process a large amount of information from different types of sensors including cameras, radar, light detection and ranging or laser imaging detection and ranging (LiDAR), ultrasound, and global positioning system (GPS), among others [15]. The information captured by the sensors must be processed quickly, so the rapid extraction of features of interest among a large amount of data is a priority, this is the principle of intelligent sensors, which pre-process the information before sending it to the central control unit [16]. This is where deep network-based models are most applicable, these can be embedded in an electronic system to form, with the appropriate instrumentation, an intelligent system [17].

Deep neural networks, and of these in particular convolutional neural networks (ConvNet), are the most advanced learning-based architectures for applications of categorizing input parameters from large numbers of learning samples [18], [19]. Training these architectures is always a challenging problem as it involves high-dimensional optimization [20]. The goal of the designer is to tune the hyperparameters of the model by minimizing the loss function (error between the model response and the actual category to which the input data belongs). This process involves the use of an optimization function, which must be selected and tuned to produce a fast fit with low memory costs and no over- or under-fitting [21]. Other problems with optimizer tuning lie in avoiding the local minima of the loss function, as well as the curse of dimensionality [22]. As a solution, different optimizers have been proposed in recent years, generally based on adaptive estimation, which can be adjusted to a large number of problems. The final choice between one optimizer over another will depend on the characteristics of the database used to train the model [23].

Many researchers and developers of deep neural models tend to select a priori the adaptive moment estimation (Adam) optimizer as the optimizer for most cases due to its high performance, particularly when working with sparse databases. However, it has been shown that Adam (as well as its variants) can become unstable under certain conditions, producing models with lower generalization capability, so it is not a generalized solution for every optimization problem [24]. The final selection of an optimizer for a given problem requires an analysis of the performance of all possible options according to the neural model to be trained and the characteristics of the dataset to be identified. This research is a work in this direction in the search for an intelligent sensor system for autonomous driving applications. The residual neural network (ResNet) type neural network has been selected for the model due to its high performance and reduced architecture compared to structures such as dense convolutional network (DenseNet) and neural architecture search network (NASNet) [25].

2. PROBLEM FORMULATION

Gradient descent is the most widely used optimization method in machine learning algorithms and neural networks [26]. In its simplest form, this algorithm seeks to minimize an objective function called $J(\theta)$ according to the θ parameters of a model. Where $\theta \in \mathbb{R}^d$, by updating the value of these parameters in the opposite direction of the gradient of the objective function, i.e.:

$$\nabla_\theta J(\theta) \tag{1}$$

The rate of change of the model in response to the error at each update step is controlled by a parameter called the learning rate. In this way, the algorithm moves following the direction of the slope of the error surface (loss function) created by the objective function downhill until it reaches a valley. The accuracy and update time of the algorithm depend inversely on the amount of data in each update. This is a parameter that differentiates the variants of gradient descent.

In the traditional gradient descent scheme (also known as batch gradient descent or vanilla gradient descent) the gradient is calculated as the average gradient for the entire training data set, i.e., the gradient is calculated as the mean of the gradient for the entire training data set:

$$\nabla f = \frac{1}{n}\sum_i \nabla loss\left(x_i\right) \tag{2}$$

because the gradients of the entire data set are computed to perform each update, this scheme tends to be very slow and impossible to use for large data sets in memory-constrained systems. This algorithm converges to the global minimum for convex error surfaces and a local minimum for non-convex surfaces.

In the case of stochastic gradient gescent (SGD), this optimizer performs the parameter update for a sample of the data set. For example, with a batch size of one, the gradient can be estimated as (3).

$$\nabla f \approx \nabla loss\left(x^{*}\right) \tag{3}$$

Where x^{*} is a random sample of the entire data set. This is therefore a variant of gradient descent that reduces the amount of data in favor of the update time. It should be noted that SGD also has many variants, such as SGD with momentum.

Gradient descent performs redundant calculations for large data sets. In these data sets, there are often similar samples, so the algorithm spends time on redundant computations. SGD eliminates this redundancy by performing a single calculation per update. The algorithm is usually faster, and in some cases can be used for online learning. It has been shown that when the learning rate is reduced the SGD shows the same convergence behavior as gradient descent. However, like gradient descent, SGD often has difficulty escaping saddle points, in these cases optimizers such as adaptive gradient (Adagrad), adaptive learning rate (Adadelta), root mean square propagation (RMSProp), and Adam generally perform much better. Adagrad is a gradient-based optimization algorithm that adapts the learning rate to the parameters in coherence with the information contained in them [27]. It tries to make smaller updates (i.e., low learning rates) for parameters associated with frequently occurring features (parameters with similar information), and larger updates (i.e., high learning rates) for parameters associated with infrequent or sparse features, so in problems where little data contains critical information the Adagrad optimizer turns out to be of great importance [28].

Since Adagrad uses different learning rates for each parameter, then one can define the gradient at time step *t* for the parameter θ_i as (4).

$$\nabla f_{t,i} = \nabla loss\left(\theta_{t,i}\right) \tag{4}$$

Adagrad calculates the learning rate at each time step *t* for each parameter according to the gradient passed for the same parameter. Adadelta is an optimizer that modifies Adagrad in terms of the behavior of this dynamic learning rate. Adadelta reduces the way this learning rate decreases monotonically without accumulating the past values of the gradient (it does not store past gradients) but limits it to a fixed value [29]. The RMSprop optimizer was developed with the same objective but with a different decay rate. Adadelta and RMSProp work almost similarly, with the only difference being that in Adadelta no initial learning constant is required.

Adam is one of the most recent optimizers, it was proposed in 2014, and like the previous cases, it is an algorithm that computes adaptive learning rates for each parameter [30]. Adam controls the decay rate similarly to Adadelta and RMSprop, but also maintains an exponentially decreasing average of past gradients similar to momentum. In simplified form, Adam can be described as behaving like the RMSprop optimizer with momentum, while nesterov-accelerated adaptive moment estimation (Nadam) is Adam as RMSprop with Nesterov momentum (Nesterov accelerated gradient). Adam has set a landmark in terms of deep learning optimizers, to the point that many researchers recommend it as a de facto solution, it combines the best features of Adadelta and RMSprop and therefore tends to perform better in most cases. However, the Adam optimizer can sometimes be unstable and is not easy to control, which is why in these cases it is suggested to use SGD with some method of learning rate update.

Adam's update rule includes the gradient of the current cycle to the power defined by the Kingma norm. High values of this norm tend to generate unstable results, so in 2015 the same authors of Adam proposed AdaMax, an optimizer that generates more stable convergences [30]. This is because the term corresponding to the current cycle gradient is essentially ignored when it is small. Thus, parameter updates are influenced by fewer gradients and are therefore less susceptible to gradient noise. In practical terms, it can be argued that AdaMax should perform better than Adam when the dataset contains sparse parameter updates, i.e., embedded information. However, in practice, this is not always true.

There is no defined strategy for the selection of a particular optimizer when approaching a deep learning model training problem. A low initial learning rate with Adam and its variants does not guarantee algorithm stability in all cases, and many solutions end up opting for classical structures such as SGD and Adadelta. Consequently, the selection of one optimizer over another should be made as a specific case according to the performance of each algorithm on the particular problem, with consideration of the degree of dispersion of the dataset. This research seeks to evaluate the best optimizer for the future design of an intelligent sensor capable of correctly and incorrectly identifying the position of a vehicle in the lane of the road (autonomous vehicle prototype).

3. RESEARCH METHOD

According to previous studies, the convolutional neural network ResNet is used as the architecture for the autonomous driving system. This architecture has been used in similar tasks and has demonstrated higher performance in image categorization, and a smaller size compared to schemes such as DenseNet and NASNet. This model uses the concept of residual connections, which means that each one (or more layers) generates a jump or forward connection from the input of the layer to the output of the same, to use the resulting sum as a new output, which is of the form $f(x) + x$. This simplifies the structure of the network and increases its learning capacity. We use the ResNet-50 variant (50 layers deep) that uses the structural block in Figure 1. This block consists of two 1×1 convolution layers to reduce the dimension at the beginning and end of the block and an inner 3×3 convolution layer.

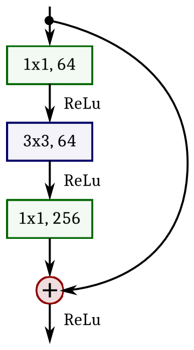


Figure 1. ResNet-50 topology structural block

The dataset used is composed of images corresponding to the open-source database PandaSet by Hesai and Scale AI [31]. This dataset is composed of LiDAR sensors and photographs from different points for vehicles on California roads. We have used for this study images corresponding to the front camera of the vehicle to build a reduced dataset with 11 characteristic locations of the vehicle, labeled from zero to 10 as:

– Category 0 - Right lane, daytime, green light
– Category 1 - Right lane, daytime, red light
– Category 2 - Left lane, daytime, green light
– Category 3 - Center lane, daytime, green light
– Category 4 - Right lane, daytime, green light, sun in front of the car
– Category 5 - Two-way road, daytime
– Category 6 - Left lane, overnight, green light
– Category 7 - Right lane, overnight, green light
– Category 8 - Center lane, overnight, green light
– Category 9 - Two-way road, overnight
– Category 10 - Center lane, daytime, red light

In each category, 100 images were placed, for a total of 1,100 images. The images were taken directly from the database without any manipulation. However, in each of the trained models, these images were randomly shuffled to avoid bias. Also, all images were normalized to a size of 256×256 pixels. Figure 2 shows an example of the images within each of these categories. We studied the efficacy of eight (8) popular stochastic gradient based optimization algorithms used in deep learning. The optimizers we compared were: i) SGD with momentum, ii) RMSprop, iii) Adagrad, iv) Adadelta, v) Adam, vi) Generalisation of Adam (Adamax), vii) Nadam, and viii) follow the regularized leader (Ftrl).

To guarantee the comparison results we kept the same hyperparameters settings for all experiments. All models were trained from scratch over 20 epochs, and categorical cross-entropy was used as the loss function. Table 1 shows the summary of the ResNet architecture used in all trained models. The models were developed in Python with Keras, Numpy, Scipy, OpenCV, Pandas, and Matplotlib. The environment used in the tests was a Linux machine with kernel 5.8.0-48-generic, Intel© Core™ i7-7700HQ CPU @2.80 GHz $\times4$ and 12 GB of RAM.

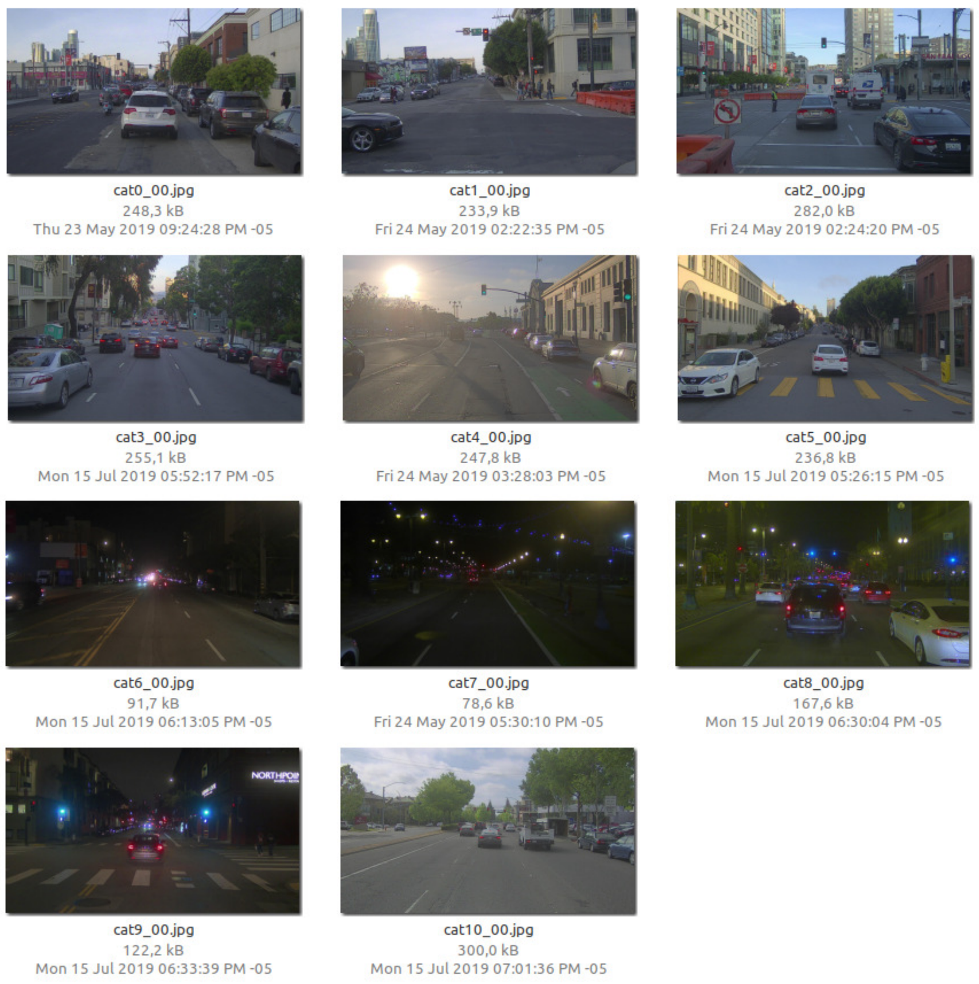

cat0_00.jpg
248,3 kB
Thu 23 May 2019 09:24:28 PM -05
cat1_00.jpg
233,9 kB
Fri 24 May 2019 02:22:35 PM -05
cat2_00.jpg
282,0 kB
Fri 24 May 2019 02:24:20 PM -05
cat3_00.jpg
255,1 kB
Mon 15 Jul 2019 05:52:17 PM -05
cat4_00.jpg
247,8 kB
Fri 24 May 2019 03:28:03 PM -05
cat5_00.jpg
236,8 kB
Mon 15 Jul 2019 05:26:15 PM -05
cat6_00.jpg
91,7 kB
Mon 15 Jul 2019 06:13:05 PM -05
cat7_00.jpg
78,6 kB
Fri 24 May 2019 05:30:10 PM -05
cat8_00.jpg
167,6 kB
Mon 15 Jul 2019 06:30:04 PM -05
cat9_00.jpg
122,2 kB
Mon 15 Jul 2019 06:33:39 PM -05
cat10_00.jpg
300,0 kB
Mon 15 Jul 2019 07:01:36 PM -05

Figure 2. Sample of the images within the 11 categories

Table 1. Summary of the ResNet architecture used in the tests

Input Layer: 256×256×3
conv1 (7×7)
Batch Normalization
ReLu
Max Pooling (3×3)
conv2 Block × 3
conv3 Block × 4
conv4 Block × 6
conv4 Block × 3
Average Pooling: 2,048
Softmax: 11
Total params: 23,610,251
Trainable params: 23,557,131
Non-trainable params: 53,120

The training in each case is performed with 80% of the images, and the remaining 20% is used for validation. The images are always randomly mixed with different seeds for each model. During training, the loss function, accuracy, and mean squared error (MSE) are calculated at each epoch for both training and validation images. These metrics will be used to evaluate the performance of each optimizer. Precision, Recall, and F1-score metrics are also calculated for the validation data, parameters that are also considered in the comparison.

4. RESULTS AND DISCUSSION

The version of SGD we used is known as Vanilla. The optimization used the basic algorithm with a constant learning rate of 0.01, a momentum of zero, and without applying Nesterov momentum. Something similar was done with the other optimizers, in none of them momentum was used, and in all of them, the same learning rate of 0.01 was used. In all cases this value can be adjusted throughout the training, however, it was decided to perform a basic configuration of all the optimizers to compare their capacity and performance with the dataset used. For the same reason, all training was performed for only 20 epochs.

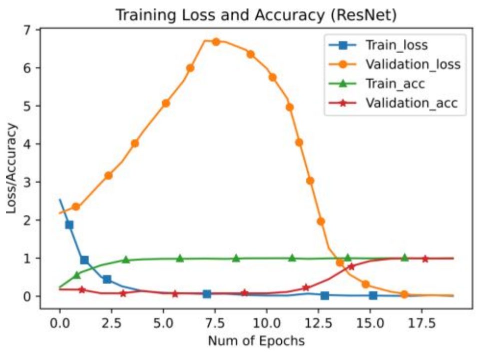
For each of the final models, Precision, Recall, and F1-Score metrics were calculated for the validation images as shown in Table 2. These data allow us to assess the final model performance with unknown data, but do not reflect the behavior of each optimizer throughout the training. To assess this aspect, during the training of the models, the loss function and accuracy were calculated for each epoch for both training and validation data, this information is shown in Figure 3. Figure 3(a) shows the behavior throughout training for the SGD optimizer, Figure 3(b) the corresponding one for the RMSprop function, Figure 3(c) for the Adagrad function, Figure 3(d) for the Adadelta function, Figure 3(e) for the Adam function, Figure 3(f) for the Adamax function, Figure 3(g) for the Nadam function, and Figure 3(h) for the Ftrl function. The objective is to observe the behavior of each optimizer throughout the training, and its ability to reduce the error of the validation data compared to its final behavior.

For the training process, Adagrad outperformed the other optimizers for the dataset of interest, followed closely by SGD. Adagrad obtained the lowest loss with the training data (less than 0.2%), and the second-lowest loss for the validation data (less than 5%), and its Accuracy for both training and validation data was almost perfect (close to 100%). This behavior was closely followed by SGD, which obtained a loss with training data below 1%, with validation data below 3%, and also an almost perfect Accuracy. Among the other optimizers, behind Adagrad and SGD, it is worth mentioning Adamax with Accuracy higher than 97% for both training and validation, and with low loss in the two groups of images (around 10%), and Adam with a little more than 95% Accuracy in the training data and 61% in the validation data, maintaining a low loss for the training data (less than 20%). Optimizers such as Adadelta, RMSprop, and Nadam performed well on the training data (Accuracy in the order of 90% and low loss), particularly Adadelta (loss less than 3%), but their performance on the validation data was very poor (about 30% Accuracy for Adadelta and RMSprop, and just below 50% for Nadam). Finally, the worst performance with this dataset was obtained with the Ftrl optimizer, which failed to reduce its loss with training data and much less with validation data.

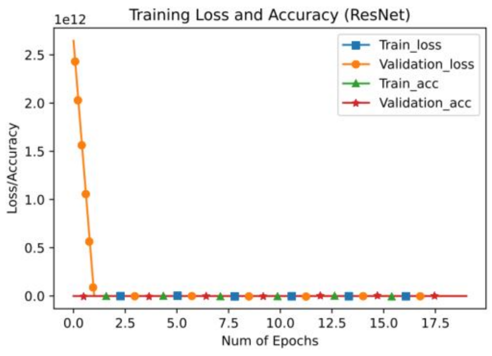
The Precision, Recall, and F1-Score metrics show similar behavior to that detected throughout the training. Again the best performance for the whole dataset (in this case considering only images from the validation group) was for the Adagrad and SGD optimizers. In both cases, these optimizers managed to correctly categorize most of the unknown images to the models. Precision provides a measurement value related to the accuracy of the model, with the number of images placed in a category that belongs to it relative to the total number of images placed within that category. On the other hand, Recall weights the sensitivity of the classification model by giving a value indicating how many of the items classified in the category belong to it. F1-Score is a weighted value of these two metrics, so it considers both false positives and false negatives in the performance assessment. Adagrad and SGD achieve 99% on all three metrics. Behind, but very close to these two optimizers is Adamax, which achieves 98% on all three metrics. Following but far behind is Adamax with 75% in Precision, 61% in Recall, and 61% in F1-Score. The other optimizers show very poor values in these metrics (below 50%), again with FTRL in the lowest place.

Table 2. Comparative results of the eight models. The worst performing metrics are highlighted in red, the best performing metrics are highlighted in green

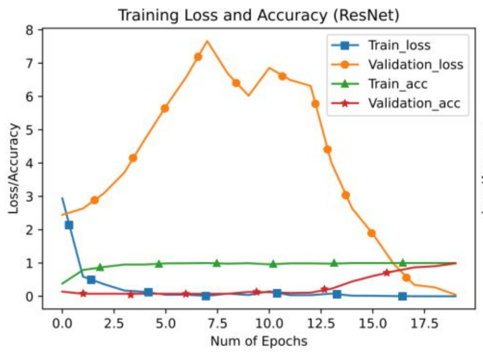
Optimizer	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy	Validation Precission	Validation Recall	Validation F1-Score
SGD	0.0067	0.0249	1.0000	0.9909	0.99	0.99	0.99
RMSprop	0.2927	45.2431	0.8967	0.3091	0.41	0.31	0.24
Adagrad	0.0017	0.0474	1.0000	0.9909	0.99	0.99	0.99
Adadelta	0.0295	6.2389	1.0000	0.3000	0.14	0.30	0.19
Adam	0.1615	2.3984	0.9535	0.6136	0.75	0.61	0.61
Adamax	0.1000	0.1063	0.9715	0.9773	0.98	0.98	0.98
Nadam	0.5207	4.7732	0.8493	0.4818	0.43	0.48	0.40
Ftrl	2.3979	2.3990	0.1016	0.0500	0.00	0.05	0.00



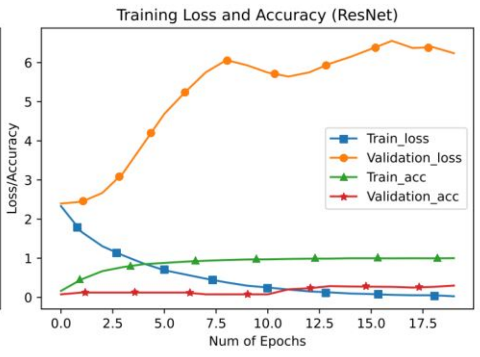
(a)



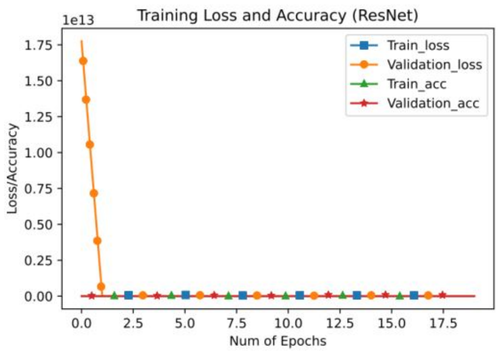
(b)



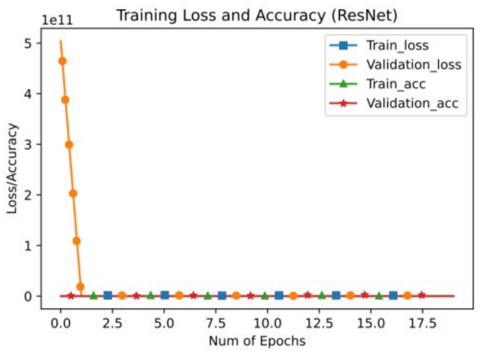
(c)



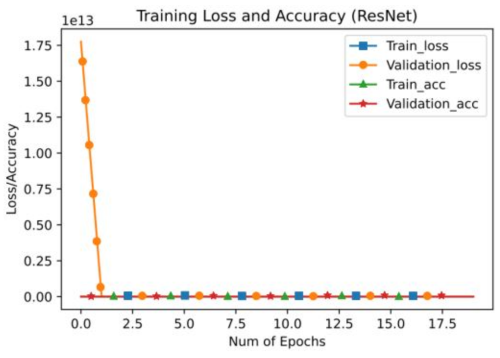
(d)



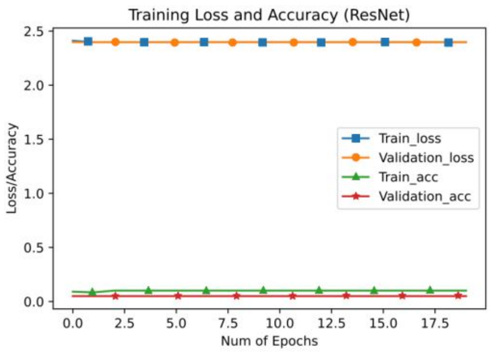
(e)



(f)



(g)



(h)

Figure 3. Behavior of the loss function and the Accuracy of the optimizer throughout the training for training and validation data (a) SGD, (b) RMSprop, (c) Adagrad, (d) Adadelta, (e) Adam, (f) Adamax, (g) Nadam, and (h) Ftrl

Figure 3 also shows that the best performance throughout the training was shown by the optimizers Adagrad, SGD, and Adamax. Unlike the latter, the other optimizers fail to reduce the loss for the validation data. The curves corresponding to RMSprop, Adam, and Nadam do not show much detail in this respect due to the scale, but Table 2 does show the poor final performance at these values. To reach a conclusion regarding the best optimizer, we identify which of the three chosen optimizers reduces the loss the fastest, which is why Figure 4 shows a detail of this behavior during the final stage of the training. Here it is observed that Adamax, despite not having a smooth decreasing behavior, manages to reduce the loss value faster than Adagrad and SGD. It is also observed that SGD has the best decreasing trend.

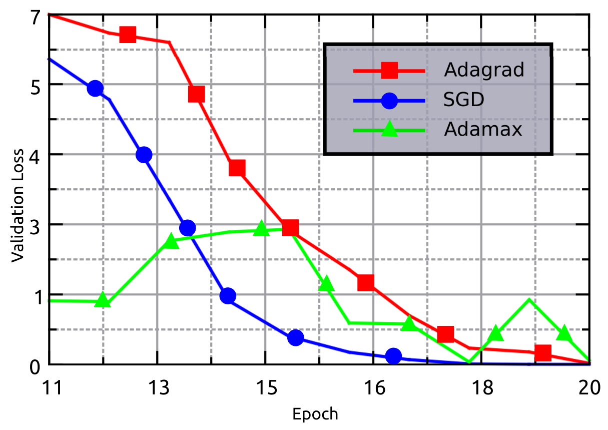


Figure 4. Detail of the behavior of the loss function of the validation data in the last training epochs for the best performing optimizers: Adagrad, SGD, and Adamax

5. CONCLUSION

This paper shows a performance comparison of eight optimization algorithms used in the fitting of a convolutional model for image classification in tasks related to autonomous driving. The research has the purpose of identifying high-performance structures for the development of intelligent sensors for autonomous driving systems, which is why different parameters of the models are evaluated, in particular, the capability of different optimizers on deep topologies in image classification tasks related to the autonomous identification of the environment by a vehicle. We use the open-source database PandaSet by Hesai and Scale AI to build a custom dataset with 1,100 images uniformly distributed in 11 categories. These categories correspond to typical locations of a car on a public road, in our case on California roads. A ResNet with 50 layers of depth was used as a deep network, and categorical cross-entropy was used as a loss function in all cases, training each model over 20 epochs. Our study showed that the performance of each optimizer is affected by the characteristics of our dataset. Recent schemes that were rated as excellent performers such as Adam were strongly outperformed by classics such as SGD. From the results, Adagrad, SGD, and Adamax outperformed the other optimizers in performance, particularly when evaluating their behavior with images not used in the training. It should be noted that no convergence acceleration strategy such as momentum and nesterov acceleration gradient (NAG) was used to identify the structure with the highest improvement capability. In principle, the SGD optimizer is selected as the first option for the development of an automatic embedded system for scenario identification, but first, the three models will undergo fine-tuning with a larger dataset, activities that are planned as a continuation of the research.

ACKNOWLEDGEMENT

This work was supported by the Universidad Distrital Francisco José de Caldas, in part through CIDC, and partly by the Facultad Tecnológica. The views expressed in this paper are not necessarily endorsed by Universidad Distrital. The authors thank the research group ARMOS for the evaluation carried out on prototypes of ideas and strategies.

REFERENCES

[1] J. Ni, T. Gong, Y. Gu, J. Zhu, and X. Fan, "An improved deep residual network-based semantic simultaneous localization and mapping method for monocular vision robot," *Computational Intelligence and Neuroscience*, vol. 2020, pp. 1–14, Feb. 2020, doi: 10.1155/2020/7490840.

[2] S. F. Martínez, F. H. Martínez, and H. Montiel, "Hybrid free-obstacle path planning algorithm using image processing and geometric techniques," *ARPN Journal of Engineering and Applied Sciences*, vol. 14, no. 18, pp. 3135–3139, 2019.

[3] F. Martinez, E. Jacinto, and F. Martinez, "Obstacle detection for autonomous systems using stereoscopic images and bacterial behaviour," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 2, pp. 2164-2172, Apr. 2020, doi: 10.11591/ijece.v10i2.pp2164-2172.

[4] T. Gill, "Blame it on the self-driving car: how autonomous vehicles can alter consumer morality," *Journal of Consumer Research*, vol. 47, no. 2, pp. 272–291, Aug. 2020, doi: 10.1093/jcr/ucaa018.

[5] E. Awad *et al.*, "Reply to: life and death decisions of autonomous vehicles," *Nature*, vol. 579, no. 7797, pp. 3–5, Mar. 2020, doi: 10.1038/s41586-020-1988-3.

[6] R. Emuna, A. Borowsky, and A. Biess, "Deep reinforcement learning for human-like driving policies in collision avoidance tasks of self-driving cars," *arXiv preprint arXiv:2006.04218*, Jun. 2020.

[7] H. M. Thakurdesai and J. V. Aghav, "Autonomous cars: technical challenges and a solution to blind spot," in *Advances in Intelligent Systems and Computing*, vol. 1086, 2021, pp. 533–547, doi: 10.1007/978-981-15-1275-9_44.

[8] S. Coicheci and I. Filip, "Self-driving vehicles: current status of development and technical challenges to overcome," in *IEEE 14th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, May 2020, pp. 255–260, doi: 10.1109/SACI49304.2020.9118809.

[9] W. Larson and W. Zhao, "Self-driving cars and the city: Effects on sprawl, energy consumption, and housing affordability," *Regional Science and Urban Economics*, vol. 81, Mar. 2020, doi: 10.1016/j.regsciurbeco.2019.103484.

[10] G. Guidi, A. M. Lekkas, J. E. Stranden, and J. A. Suul, "Dynamic wireless charging of autonomous vehicles: small-scale demonstration of inductive power transfer as an enabling technology for self-sufficient energy supply," *IEEE Electrification Magazine*, vol. 8, no. 1, pp. 37–48, Mar. 2020, doi: 10.1109/MELE.2019.2962888.

[11] D. Hopkins and T. Schwanen, "Talking about automated vehicles: What do levels of automation do?," *Technology in Society*, vol. 64, Feb. 2021, doi: 10.1016/j.techsoc.2020.101488.

[12] C. Jung, D. Lee, S. Lee, and D. H. Shim, "V2X-communication-aided autonomous driving: system design and experimental validation," *Sensors*, vol. 20, no. 10, May 2020, doi: 10.3390/s20102903.

[13] A. Rendón, "Operational amplifier performance practices in linear applications," *Tekhnê*, vol. 16, no. 1, pp. 57–68, 2019.

[14] B. Osinski *et al.*, "Simulation-based reinforcement learning for real-world autonomous driving," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 6411–6418, doi: 10.1109/ICRA40945.2020.9196730.

[15] J. Tang, S. Liu, L. Liu, B. Yu, and W. Shi, "LoPECS: a low-power edge computing system for real-time autonomous driving services," *IEEE Access*, vol. 8, pp. 30467–30479, 2020, doi: 10.1109/ACCESS.2020.2970728.

[16] G. Chen, H. Cao, J. Conradt, H. Tang, F. Rohrbein, and A. Knoll, "Event-based neuromorphic vision for autonomous driving: a paradigm shift for bio-inspired visual sensing and perception," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 34–49, Jul. 2020, doi: 10.1109/MSP.2020.2985815.

[17] A. Rendón, "Design and evaluation of volume unit (VU) meter from operational amplifiers," *Tekhnê*, vol. 16, no. 2, pp. 31–40, 2019.

[18] F. Martinez, C. Hernández, and A. Rendón, "Identifier of human emotions based on convolutional neural network for assistant robot," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 3, pp. 1499-1504, Jun. 2020, doi: 10.12928/telkomnika.v18i3.14777.

[19] Y. Li *et al*., "Deep learning for LiDAR point clouds in autonomous driving: a review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3412–3432, Aug. 2021, doi: 10.1109/TNNLS.2020.3015992.

[20] M. Geiger, S. Spigler, A. Jacot, and M. Wyart, "Disentangling feature and lazy training in deep neural networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2020, no. 11, Nov. 2020, doi: 10.1088/1742-5468/abc4de.

[21] K. S. Chahal, M. S. Grover, K. Dey, and R. R. Shah, "A hitchhiker's guide on distributed training of deep neural networks," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 65–76, 2020, doi: 10.1016/j.jpdc.2019.10.004.

[22] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen, "A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations," *SN Partial Differential Equations and Applications*, vol. 1, no. 2, Apr. 2020, doi: 10.1007/s42985-019-0006-9.

[23] M. H. Saleem, J. Potgieter, and K. M. Arif, "Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers," *Plants*, vol. 9, no. 10, pp. 1–17, 2020, doi: 10.3390/plants9101319.

[24] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, and W. E, "Towards theoretically understanding why SGD generalizes better than ADAM in deep learning," *Advances in Neural Information Processing Systems*, Oct. 2020.

[25] E. Jeong, Y. Suh, and D. Kim, "A study on the outlet blockage determination technology of conveyor system using

deep learning," *Journal of the Korea Society of Computer and Information*, vol. 25, no. 5, pp. 11–18, 2020, doi: 10.9708/jksci.2020.25.05.011.
[26] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, Sep. 2016.
[27] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.
[28] J. Dean *et al.*, "Large scale distributed deep networks," *Advances in Neural Information Processing Systems*, vol. 2, pp. 1223–1231, 2012.
[29] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, Dec. 2012.
[30] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," *3rd International Conference on Learning Representations*, pp. 1–15, Dec. 2014.
[31] "Scale and Hesai's PandaSet." scale.com. https://scale.com/open-datasets/ pandase (accessed Feb. 10, 2021).

BIOGRAPHIES OF AUTHORS

Fernando Martínez is a doctoral researcher at the Universidad Distrital Francisco José de Caldas focusing on the development of navigation strategies for autonomous vehicles using hierarchical control schemes. In 2009 he completed his M.Sc. degree in Computer and Electronics Engineering at Universidad de Los Andes, Colombia. He is a researcher of the ARMOS research group (Modern Architectures for Power Systems) supporting the lines of electronic instrumentation, control and robotics. He can be contacted at email: fmartinezs@udistrital.edu.co.

Holman Montiel is a professor of algorithms, embedded systems, instrumentation, telecommunications, and computer security at the Universidad Distrital Francisco José de Caldas (Colombia) and a researcher in the ARMOS research group (Modern Architectures for Power Systems). His research interests are encryption schemes, embedded systems, electronic instrumentation, and telecommunications. Montiel holds a master's degree in computer security. He can be contacted at email: hmontiela@udistrital.edu.co.

Fredy Martínez is a professor of control, intelligent systems, and robotics at the Universidad Distrital Francisco José de Caldas (Colombia) and director of the ARMOS research group (Modern Architectures for Power Systems). His research interests are control schemes for autonomous robots, mathematical modeling, electronic instrumentation, pattern recognition, and multi-agent systems. Martinez holds a Ph.D. in Computer and Systems Engineering from the Universidad Nacional de Colombia. He can be contacted at email: fhmartinezs@udistrital.edu.co.