Implementation and Optimization of Image Processing Algorithm using Machine Learning and Image Compression

Georgios Zacharis, *Giannis* Gadounas, *Pashalis* Tsirtsakis, *George* Maraslidis, *Nikolaos* Assimopoulos, and *George* Fragulis[1,*]

[1]Department of Electrical and Computer Engineering, University of Western Macedonia, Kozani, Greece

Abstract. This research paper deals with the implementation of an image captioning algorithm using Tensorflow, Keras, and Python, as well as a way proposed for optimization, using image compression techniques. The objective is to use image compression techniques to minimize data size, execution time, and computer resources since machine learning applications often have numerous constraints concerning energy consumption, processing power, and dataset sizes, thus making them less efficient for real-time, applied use cases. We can find new ways to apply machine learning in more simple real-life applications by attempting to reduce such obstacles.

1 Introduction

It is known that machine learning applications face difficulties such as big dataset sizes, big power consumption, and time-consuming training times. In many cases, these obstacles are in the way of such technologies being used in real-time applications efficiently. Such problems fuel the present research paper concerning ways to attempt to reduce problems like the ones mentioned above thus making the image captioning algorithm more efficient to be used in real life and real-time applications [1–3]. The research of this paper is the optimization of a Python-based image captioning algorithm using image compression and decomposition methods so that comparison data can be extrapolated after being tested in the same set of images. The results are being also analyzed in-depth via the aid of graphs after the testing stage so that the extent of the impact is presented. It is important to note that the "extent of the impact" is measured in multiple areas like execution time, dataset size, and machine resources analyzed in detail.

2 Background

Apart from education applications [4], machine learning and Neural Networks could be used in various other such as sign language learning [5], gaming [6], and medical applications [7]. Due to the constraints mentioned before, as well as faulty or lesser quality equipment, bad conditions that can affect the hardware's performance, (for example very bright or very dark lightning conditions) the efficiency of such models may be unstable making them, for now, not efficient enough for real life use. However, there are many applications that can adapt a good and efficient image captioning model that do not necessarily require hardware of the best quality so that they can be accessible to more people. An example can be aids for disabled individuals. Such an application needs to be effective, which includes fast response, accuracy in a dynamic environment and reliability. Another use can be a smart glass application for live feed image captioning and object recognition for commercial, entertainment or educational purposes, like producing instant text for paintings in an art gallery or recognizing the model and brand of products. Such possibilities led us to attempt to optimize and possibly suggest ways to make an image captioning model more efficient.

3 Objective

3.1 Software-hardware tools

Before analyzing the process let's look at the technical parts first. The image captioning algorithm was developed in Python, Python 3 to be precise, as it is almost a main way in machine learning applications for its easily accessible features using libraries. We used the Tensorflow platform in combination with the Keras library. Also, we wanted to have as much GPU power as possible. The Google Collaboratory platform was a simple solution in this area as it provides some of the tools needed to be preinstalled, so we ended up using it for our tests. In particular, the specifications of the machine we used are shown in table 1. The specifications of the machine used are a good start so that we could extract some general results. At first, the training of the model was done using the Flickr30k Image dataset. After that, some random images were selected and a new caption was generated for each one, where then we compared the initial and the generated caption to extract the accuracy of the model. Afterward, we decided to use image compression techniques on the datasets to see the quality of the results in an attempt to reduce execution

*e-mail: gfragulis@uowm.gr

Table 1. Software tools and Hardware

Software	Hardware
Python Version 3.7.12	2x Intel Xeon 2.20 GHz CPU
Tensorflow 2.7.0	Tesla P100-PCIE-16GB GPU
Keras 2.7.0	13GB RAM
Cuda 11.1 V11.1.105	78GB Storage

time, dataset size and see the effects on the performance. The compressed images were unseen to the trained model, but the original images were part of its training. There are 2 categories of image compression techniques: lossy and lossless [8–10]. Lossless compression means that as the file size is compressed, the picture quality remains the same and it does not get worse. Also, the file can be decompressed to its original quality. Lossy techniques on the other hand permanently remove data from the original for the needs of the compression [11]. In this paper, we focus on how the image captioning algorithm operates on unseen images after lossy image compression techniques. We decided to use 3 lossy techniques PCA, SVD, and K-means.

4 Methodology

4.1 CNN-RNN

The Image Captioning Algorithm is based on Convolutional Neural Network (CNN)-Recurrent Neural Network (RNN) framework [12–16] that first extracts the image-level feature using CNN, and then utilizes a language model, RNN or its variants, such as Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) et al., to generate meaningful sentences [17]. The CNN is used to encode the image where the RNN to decode the sentence from the embedding [18].

4.2 PCA

The first image compression technique chosen is Principal Component Analysis (PCA). PCA is one of the most popular unsupervised machine learning algorithms as an analysis tool in signal processing, image processing, and pattern recognition, where it transforms several correlated variables into a smaller set of uncorrelated variables [19]. The data values of the dataset are defined as p n-dimensional vectors $x_1, \ldots x_p$ or equivalently, a $n \times p$ data matrix X, whose j^{th} column is the vector x_j of observations on the jth variable, where p numerical variables and n entities or individuals. Such dimensionality reduction can be a very useful step for visualizing and processing high-dimensional datasets, while still retaining as much of the variance in the dataset as possible[20]. Even though the technique is lossy, PCA-based dimensionality reduction tends to minimize that information loss, under certain signal and noise models. Although it seems almost like the compressed image using the PCA method it is not, as explained. Under the assumption that $x = s+n$, that is, the data vector x is the sum of the desired information equals the bearing signal s and a noise signal n one can show that

Figure 1. The original uncompressed image

Figure 2. The same image compressed with PCA algorithm

PCA can be optimal for dimensionality reduction, from an information-theoretic point-of-view.

4.3 SVD

The second compression technique chosen is Singular Value Decomposition (SVD). SVD is another dimensionality reduction algorithm, that works by rewriting the complicated original matrix into a sum of smaller and simpler matrices [21]. This computation allows us to present the substructure of the original data more accurately, releasing the values that are not as necessary in retaining the quality of the image. It also orders this data from most variation to the least variation. Specifically, the singular value decomposition of an $m \times n$ complex matrix M is a factorization of the form $M = U\Sigma V*$, where U is $m \times m$ complex unitary matrix, Σ is a $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal, and V is an $n \times n$ complex unitary matrix [22]. On the right you see an image before and after the compression using SVD.

4.4 K-Means

The final technique chosen is K-mean clustering. Because of its simplicity, k-means clustering algorithm was chosen

Figure 3. The same image compressed with SVD

as another popular method in image segmentation, with the difference that it applies cluster analysis in contrast to the previous techniques. This method aims to partion the dataset $X = x_1, \ldots, x_N$, where $x_N \in Rd$, into M subsets/clusters $C_1, \ldots, C_M$, depending on the criterion applied [23]. The sum of the squared Euclidean distances between each data point x_i and joint m_k (cluster center) of the subset C_k, is the most widely used and the one we utilized [24].

Figure 4. The same image compressed with K-Means

5 Experimental Results

5.1 Image captioning

The image captioning algorithm [25], a CNN & RNN architecture as mentioned before [26], has 8 hidden layers and runs for 1000-10000 iterations with an average execution time 45min / 1000 iterations. Of course, we are talking about GPU runtime. For the training, we used the Flickr30K dataset, which, as the name states, is a 30k images dataset. Furthermore, we experimented with values like the number of iterations, learning rate, or the number of layers so that we can find an optimal tuning. As it is shown on the images the model deals well as a comparison to the original caption of the image with the caption it predicts. There are some extra words in the predicted captions, a problem that can be resolved if the accuracy rate is improved.

Figures 5 and 6 show the average values of the algorithm during all the runs, regarding the cost and accuracy. Having experimented and extrapolated all this data we came up with an ideal tuning of parameters for the model at 8000 iterations with an accuracy percentage of 65%. Let it be noted that as shown in the graphs, the model was trained many times with different parameters, so this accuracy rate is not a result of one run but a rather stable average accuracy rate.

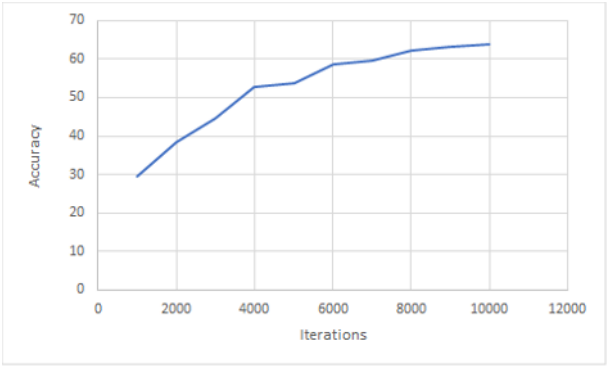

Figure 5. The average values of the algorithm's accuracy during all the runs

The results from testing and experimentation gave us a much more clear image of how the initial image captioning algorithm was effected.

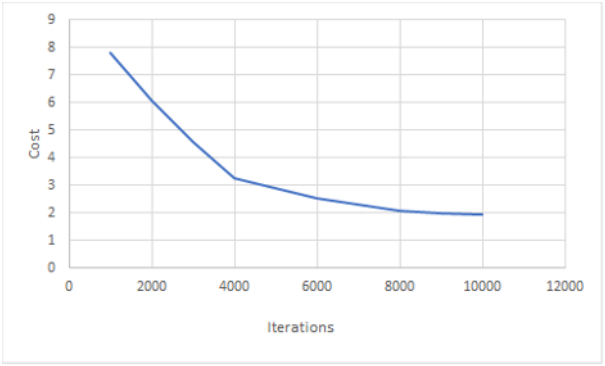

Figure 6. The average values of the algorithm's cost during all the runs

After the training with 30k images 30 random images of this dataset were taken to be compressed 3 times for each compression method and given to the model as an input for image captioning. As shown in the images below there is a big inaccuracy rate in the predicted captions but after getting all the results, we noticed that in some cases there are fractions of accuracy for the PCA and the K- means compressed images while SVD was unsuccessful.

On figure 8, the predicted captions are wrong, but the model chose right words such as "small girl" and "green" in the PCA image which is a good sign.

Figure 7. Image captioning results on original image

Figure 8. Image captioning results on PCA compressed image

Same case is presented on figure 10. As we see this time the model gave a rather more accurate prediction for the K-Means compressed image than the original image itself. Such inaccurate outputs were partly expected as the model was not trained in compressed images. How-

Figure 9. Image captioning results on original image

Figure 10. Image captioning results on K-Means compressed image

ever, such clues suggest that the algorithm can, partly for now, "see" the compressed images. The results of the calculations of the data size and execution time reduction were quite promising. Of all 3 compression techniques K-means compression method saves the most size per image at around 65% each image while others around 50%. There was also a big improvement in overall execution time. The fastest technique was PCA with approximately 0.28 sec per image while the other 2 techniques also showed execution time improvement. These percentages can make a substantial difference in using techniques like image captioning in more applied real-time applications. Such signs suggest that there might be more to this experiment, as it is also backed up by the results extrapolated. With better training of the neural network and the overcoming of some problems, image compression can be an efficient way to make the image captioning algorithm more efficient.

6 Discussion - Problems

Taking stock of the results shown above and the experience earned we faced some key obstacles in this research which we wish to solve in our future work. As a general, google collab was friendly to use environment with all the tools we needed installed by default. But there were limitations as for the disc space, GPU runtime, and background execution. Furthermore, the average execution time for training at an average of 5000 iterations is 4.30 hours, depending on the stability of the bandwidth which most of the time was not on our side. Here there is an area for improvement because in future work the algorithm needs to be trained for datasets of a bigger size. The results show that despite the low accuracy of the compression algorithms used, the dataset's size is reduced to a great extent. It is expected that if the accuracy of the image captioning algorithm is increased then we can see better results generally in many areas of the research. This leads to much work to be done for future work in our research to solve the issues mentioned.

7 Conclusions and Future Work

In conclusion, the accuracy of the model after using compression has plenty of areas for improvement. But there are good and promising signs, such as substantial data size and execution time reduction, that make us have an optimistic view about the potential of this research. With improvements in our future endeavors, this model can be efficient enough to be used in practical situations. At first, improving the image captioning algorithm by creating a different and better deep neural network architecture or by trying to imply parallelization methods into the model to improve the overall accuracy of captions and reduce the execution time. Furthermore, using a more powerful computer, with better hardware and maybe a Linux based OS (to give more freedom for tuning several parameters and values in the system) will give a chance for training and testing the model across more challenging benchmark datasets, like MSCOCO, VQA 2.0 et al., and compare the results for a clearer view of the status. At last, experimenting with different decomposition or compression techniques could make up a better model with higher accuracy and lower execution time.

References

[1] H. Diao, Y. Zhang, L. Ma, H. Lu, arXiv preprint arXiv:2101.01368 (2021)

[2] S. Ding, S. Qu, Y. Xi, A.K. Sangaiah, S. Wan, Pattern Recognition Letters **123**, 89 (2019)

[3] G. Barlas, C. Veinidis, A. Arampatzis, The Visual Computer **37**, 1309 (2021)

[4] G.F. Fragulis, M. Papatsimouli, L. Lazaridis, I.A. Skordas, Software Impacts **7**, 100046 (2021)

[5] M. Papatsimouli, L. Lazaridis, K.F. Kollias, I. Skordas, G.F. Fragulis, SHS Web Conf. **102**, 01008 (2021)

[6] L. Lazaridis, M. Papatsimouli, K.F. Kollias, P. Sarigiannidis, G.F. Fragulis, *Hitboxes: A Survey About Collision Detection in Video Games*, in *International Conference on Human-Computer Interaction* (Springer, 2021), pp. 314–326

[7] K.F. Kollias, C.K. Syriopoulou-Delli, P. Sarigiannidis, G.F. Fragulis, Electronics **10**, 2982 (2021)

[8] A. Said, W.A. Pearlman, IEEE Transactions on image processing **5**, 1303 (1996)

[9] F. Yang, J. Mou, K. Sun, R. Chu, Multimedia Tools and Applications **79**, 19963 (2020)

[10] S. Cao, C.Y. Wu, P. Krähenbühl, arXiv preprint arXiv:2004.02872 (2020)

[11] O.K. Al-Shaykh, R.M. Mersereau, IEEE Transactions on Image Processing **7**, 1641 (1998)

[12] J. Aneja, A. Deshpande, A.G. Schwing, *Convolutional image captioning*, in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 5561–5570

[13] W. Jiang, L. Ma, Y.G. Jiang, W. Liu, T. Zhang, *Recurrent fusion network for image captioning*, in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 499–515

[14] M.Z. Hossain, F. Sohel, M.F. Shiratuddin, H. Laga, ACM Computing Surveys (CsUR) **51**, 1 (2019)

[15] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, L. Zhang, *Bottom-up and top-down attention for image captioning and visual question answering*, in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 6077–6086

[16] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, Y. Bengio, *Show, attend and tell: Neural image caption generation with visual attention*, in *International conference on machine learning* (PMLR, 2015), pp. 2048–2057

[17] J. Gu, G. Wang, J. Cai, T. Chen, *An empirical study of language cnn for image captioning*, in *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 1222–1231

[18] H. Parikh, H. Sawant, B. Parmar, R. Shah, S. Chapaneri, D. Jayaswal, *Encoder-decoder architecture for image caption generation*, in *2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA)* (IEEE, 2020), pp. 174–179

[19] I.T. Jolliffe, J. Cadima, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **374**, 20150202 (2016)

[20] C. Clausen, H. Wechsler, pattern recognition **33**, 1555 (2000)

[21] H. Swathi, S. Sohini, G. Gopichand et al., *Image compression using singular value decomposition*, in *IOP Conference Series: Materials Science and Engineering* (IOP Publishing, 2017), Vol. 263, p. 042082

[22] E.A. Compton, S.L. Ernstberger (????)

[23] A. Likas, N. Vlassis, J.J. Verbeek, Pattern recognition **36**, 451 (2003)

[24] G. Hamerly, C. Elkan, *Alternatives to the k-means algorithm that find better clusterings*, in *Proceedings of the eleventh international conference on Information and knowledge management* (2002), pp. 600–607

[25] G. Zacharis, G. Gadounas, P. Tsirtsakis, *Image captioning and image compression*, https://github.com/TechZx/Image-captioning-and-image-compression (2022)

[26] S. Bai, S. An, Neurocomputing **311**, 291 (2018)