Artificial Intelligence & Robotics Development Journal
Volume 3, Issue 4, pp 269-281, October 2023, https://doi.org/10.52098/airdj.2023348
ISSN: 2788-9696 Received: 18/12/2023 Revised: 2/1/2024 Accepted: 3/1/2024

Enhancing the Accuracy of Image Classification using Deep Learning and Preprocessing Methods

Mohammed J. Yousif[1,*], Mohammed Balfaqih[2]
[1,2] Department of Computer Science, Faculty of Science, Memorial University, Canada.

* Corresponding author: Mohammed J. Yousif [1,*], malmutoory@mun.ca

Abstract

Deep learning is one of many methods in Artificial Intelligence (AI) that computers can use to process information like text, images, and audio. This manuscript will focus on image preprocessing, one of the many different techniques used to modify the neural network model training process, and how it affects the training speed and accuracy of the neural network. Six different image preprocessing techniques were picked for use in this study: Grayscale, Smoothing, Unmask Sharpening, Laplacian and Equalization, and Random Cropping and Rotation, all of which were implemented using Python and the libraries NumPy, OpenCV, and PyTorch. For the dataset, a batch of 10000 images from the CIFAR10 dataset was used to train the model. This study explored the impact of preprocessing techniques on a deep learning model employing the RESNET50 architecture. Notable improvements in model accuracy were observed, particularly with normalization and random cropping accompanied by rotation. The efficiency gains attributed to preprocessing were highlighted, leading to a more rapid training process and significant resource savings. This research underscores the importance of thoughtful preprocessing in enhancing the performance of deep learning models, offering valuable insights for practitioners in image classification tasks.

***Keywords**:* Deep learning; Image Classification; Preprocessing Methods; neural network model; Artificial Intelligence.

1. Introduction

In recent times, computer vision has undergone a significant transformation, primarily due to the remarkable progress in deep learning. Among the different applications in computer vision, image classification is a crucial and widely used task. Image classification involves sorting visual data into pre-defined classes or labels, enabling machines to identify and distinguish objects within images. Deep learning, particularly convolutional neural networks (CNNs), has emerged as a powerful approach for image classification tasks. The capacity of deep learning models to automatically learn hierarchical features from raw pixel data has led to unprecedented accuracy and efficiency in image recognition. This paradigm shift has had a significant impact on various domains, including healthcare, autonomous vehicles, security, and many others, where precise and rapid image classification is crucial.

Deep learning is one of many methods in Artificial Intelligence (AI) that computers can use to process information like text, images, and audio (Alkishri et al., 2023). It has been around for a very long time, since the 1940s, and they have gone through several different phases which has produced a variety of different architectures such as FFNN (Feed Forward Neural Networks), CNN (Convolutional Neural Network) and RNN (Recurrent Neural Networks) (Khamis & Yousif, 2022). Evolution strategies (ES) and advanced preprocessing techniques complement each other in enhancing the accuracy of neural networks (Lapid & Sipper, 2022). Although neural networks are robust in learning intricate representations from data, their performance relies heavily on the preprocessing of input data. Integrating evolution strategies into the training process introduces an adaptive and evolutionary dimension, which boosts the neural network's ability to learn and generalize effectively. Data preprocessing is vital for neural network techniques to refine raw data and improve learning for better model performance (Zhou et al., 2023). Scaling input features through normalization and standardization, generating additional training samples through data augmentation, filtering noise through noise reduction, and feature engineering can significantly boost the neural network's performance. However, throughout those phases, there was also an evolution in strategies to improve the training outside of changing/improving the architecture itself, such as data augmentation, early stopping, transfer learning, hyperparameter tuning and image preprocessing. The one method that stands out is image preprocessing since the quality of the input data is just as crucial as the quality of the model itself (Hasoon et al., 2011). No matter how good the model is, if the input data isn’t any good then the results won’t be either, the saying “garbage in, garbage out” rings a bell here. With the exponential increase in size of neural networks and their processing and data requirements, there is an ever-increasing need for optimisation of the data itself. Figure 1 shows that there is an exponential increase

in the number of research papers focusing on image preprocessing for neural networks per year. This paper will be focusing on evaluating several different preprocessing methods and combinations of them to see if they can improve the training time and training accuracy of the neural network model. Also, a few different models will be evaluated to see which ones fit the purpose of this manuscript best, which is the classification of the CIRAF10 dataset.

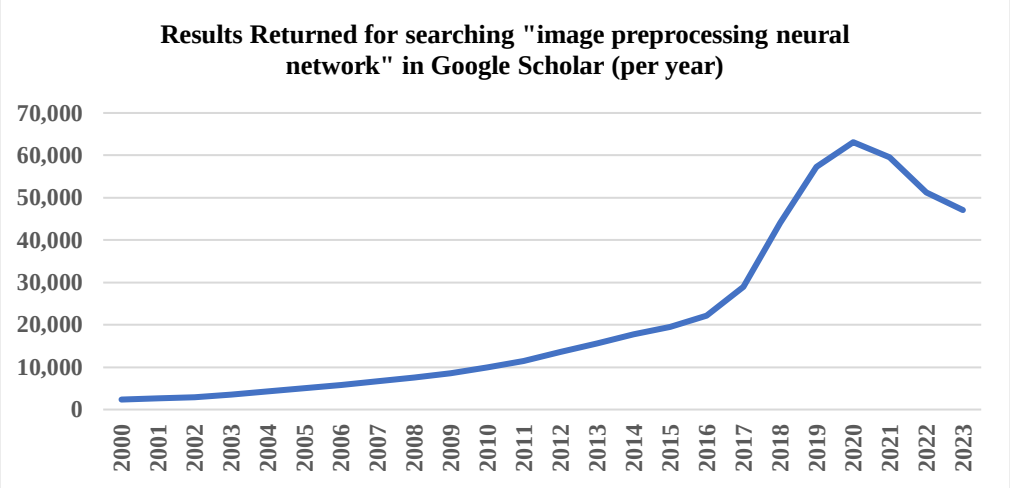

Figure 1: Results Returned for searching "image preprocessing neural network" in Google Scholar

2. Literature Survey

Many Many studies examined image classification using different preprocessing and neural network techniques. Ghandour (Ghandour et al., 2023) investigated the effectiveness of a convolutional neural network (CNN) model for medical image classification and extraction. They generated consistent feature maps for deep learning-based medical image fusion and tested the proposed model using various medical imaging methods. The results showed that the model achieved better image diagnosis and competing quantitative metrics. In 2023, Yogeshwari et al. developed a neural model to detect plant leaf diseases using deep convolutional neural networks (DCNN). They used various preprocessing methods and filtering techniques, including a 2D Adaptive Anisotropic diffusion filter and some enhancement techniques. They also implemented a clustering method based on the Improved Fast Fuzzy C Means approach. According to their results, the proposed framework was more effective than other classifiers and achieved better classification results (Yogeshwari & Thailambal, 2023). Şengöz (Şengöz et al., 2022) proposed the use of CLAHE (Contrast Limited Adaptive Histogram Equalization) to process the images before training the Deep Convolutional Neural Network on them. They found that there was an improvement of 5% for the F1 score (from 93% to 98%) for the neural network that was trained on the CLAHE processed images.

Öztürk et al. (2018) performed a review study to compare the effect of different levels of Histopathological processing on the results of a Convolutional Neural Network, with each level adding on extra processing on the previous level's results. Their results showed that the normal level of preprocessing performed best. This is because using too much preprocessing ended up removing important details from the input images that are useful for the neural network (Öztürk et al., 2018). Calderon et al. (2018) performed a study to test the effect of using Deceived Non-Local Means (DNLM) filter as a preprocessing step on a Convolutional Neural Network that is designed to estimate ages. It was found that using this method gave a 42% lower root mean squared error compared to just using the original data (Calderon et al., 2018). Atomi (2012) ran a review to determine if the use of data preprocessing techniques is effective for Artificial Neural Networks. Min-max normalization, Z-Score normalization and decimal scaling normalization were all used in this study. It was found that there was a significant improvement in the efficiency of the training and performance of the ANN (Atomi W, 2012). Tabik et al. (2017) reviewed the use of different image preprocessing techniques on three different CNN based neural networks (LeNet, Network3 and DropConnect). The methods that were used for the preprocessing were centering, elastic deformation, translation, rotation, and different combinations of them. In terms of accuracy, most methods had very similar scores to original image (within margin of error). However, when it comes to training time there was a sizable improvement when using centered image preprocessing (from 270 to 200 seconds for example) (Tabik et al., 2017). Table 1presents a summary of literature survey studies. The studies indicate different gaps in evaluating metrics of CNN models, so there is a need for standardized evaluation metrics. Investigating the generalizability of preprocessing techniques and CNN architectures across diverse domains could provide valuable insights. Understanding the impact of data characteristics on selecting preprocessing techniques is crucial. A comprehensive investigation into the robustness of CNNs to noisy data is warranted. There needs to be more guidance on selecting optimal preprocessing strategies. Exploring hybrid approaches could lead to more robust preprocessing pipelines. Understanding how preprocessing impacts the interpretability of CNN models is crucial. A more comprehensive exploration of the computational efficiency of preprocessing techniques is needed. A meta-analysis or systematic review can provide a comprehensive overview of the current knowledge in image preprocessing for CNNs.

3. Preprocessing Methods

The implementation for the preprocessing code was written in the Python programming language (Pajankar & Joshi, 2022). Three libraries were used: Math (for its sin and cos functions), OpenCV (for importing images and

creating black bordering around images) and NumPy (for manipulating the images as 2D arrays). Eight different preprocessing methods were proposed and implemented for the purpose of this manuscript: Grayscale, Smoothing, Unmask Sharpening, Laplacian, Equalization, Random Rotation, Random Cropping and Normalization. For all the above methods, different loops were used to run through each pixel (or even each RGB channel) and the preprocessing method was performed on each one (Al-Hatmi & Yousif, 2017).

Table 1: Summary of literature survey studies.

Author	Preprocessing Method	Neural Network Model	Results
Ghandour et al., 2023	Different processing methods	Convolutional Neural Network	generate consistent feature maps; better image diagnosis and competing quantitative metrics.
Yogeshwari et al., 2023	various preprocessing methods; filtering techniques	Deep convolutional neural networks + Fuzzy C Means approach	More effective than other classifiers.
Şengöz et al., 2022	CLAHE (Contrast Limited Adaptive Histogram Equalization)	Deep Convolutional Neural Network	5% improvement in F1 score (from 93% to 98%) for neural network trained on CLAHE processed images
Öztürk et al., 2018	Different levels of Histopathological processing	Convolutional Neural Network	Best performance with normal preprocessing; excessive preprocessing led to the removal of important details from input images
Calderon et al., 2018	Deceived Non-Local Means (DNLM) filter	Convolutional Neural Network (for age estimation)	42% lower root mean squared error compared to using original data
Tabik et al., 2017	Centering, elastic deformation, translation, rotation, and combinations	CNN based neural networks (LeNet, Network3, DropConnect)	Similar accuracy to original images, but substantial reduction in training time with centered image preprocessing
Atomi W, 2012	Min-max normalization, Z-Score normalization, decimal scaling normalization	Artificial Neural Networks (ANN)	Significant improvement in training efficiency and performance of ANN

For the grayscale filter, all the RGB values for a particular pixel are retrieved. Then, the intensity for that pixel will be calculated using the following equation (1):

$$\text{Intensity} = 0.299*R + 0.587*G + 0.114*B \quad \ldots (1)$$

For the smoothing filter, an average 3x3 box filter was used. A black border was added to the image before it was processed, so that applying the kernel at the edge pixels will not cause the program to crash. The kernel consists of a 3x3 matrix that is filled entirely with ones. Since this is an RGB image, this kernel was used for each RGB channel. The intensities are all added together, and their results are divided by nine to get the average intensity for that particular colour channel as in equation 2.

$$\text{average intensity} = \frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \dots (2)$$

For the unsharp masking filter, an image with the smoothing filter applied is first created. Since this is an RGB image, each colour channel will be operated on individually. Then, the following equation is used to calculate the new intensity: Intensity = 2 * Original – Smoothed.

For Laplacian, a black border is also added to the image since a kernel will be used. The kernel consists of a 3x3 matrix, with a "-4" in the middle and ones all around it horizontally and vertically. Since this is an RGB image, this kernel was used for each RGB channel. The intensities were all added together to get the new intensity for that particular channel as in equation (3).

$$\text{RGB} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \dots (3)$$

For equalization, the process is a little bit more involved. The image is first converted into the HSI colour space (Hue, Saturation, and Intensity). All the RGB values for a particular pixel are retrieved and the following equations are applied (4 and 5):

$$Intensity = \frac{R+G+B}{3} \qquad Saturation = 1 - \frac{3*\min(R,G,B)}{R+G+B} \quad \dots (4)$$

$$Intensity = \frac{R+G+B}{3} \qquad Saturation = 1 - \frac{3*\min(R,G,B)}{R+G+B} \quad \dots (5)$$

$$Hue = 360 - Theta \mid if\ B > G$$

This is done for each pixel, resulting in an image in the HSI colour space. After this is done, histogram equalization is performed only on the Intensity channel of this HSI image using the equation (6):

$$Intensity = (L-1) * \sum_{j=0}^{k} pr/rj \quad \dots (6)$$

Where L is the number of light levels, "k" is the total number of different unique intensities, "pr" is number of pixels of that intensity and "rj" is total number of pixels. Once this is done, then the image needs to be returned to the RGB colour space (Alighaleh et al., 2022). This is done using the functions defined in equation (7):

If 0 <= H < 120: If 120 <= H < 240: If 240 <= H < 360: … (7)

$$B = I(1 - S) \qquad R = I(1 - S) \qquad G = I(1 - S)$$
$$R = I\left(1 + \frac{S\cos H}{\cos(60° - H)}\right) \qquad G = I\left(1 + \frac{S\cos(H - 120°)}{\cos(180° - H)}\right) \qquad B = I\left(1 + \frac{S\cos(H - 240°)}{\cos(300° - H)}\right)$$
$$G = 3I - (R + B) \qquad B = 3I - (R + G) \qquad R = 3I - (G + B)$$

For random crops, assuming an image has dimensions (C, H, W), the random cropping operation involves selecting a top-left pixel position (i, j) and cropping the region of size (crop_h, crop_w). The logic can be expressed as follows:

- Randomly select 'i' from the range [0, H - crop_h]
- Randomly select 'j' from the range [0, W - crop_w]
- Cropped region = original_image[:, i:i+crop_h, j:j+crop_w]

Continuing with random rotation:

For random rotation, we consider the cropped region obtained from the previous step. The logic for random rotation can be described as follows:

- Randomly select an angle 'theta' for rotation.
- Apply rotation to the cropped region using an appropriate rotation function.

The resulting rotated image will be part of the dataset for training or validation. This process introduces variability and augmentation to the dataset, enhancing the model's ability to generalize to different orientations and positions of objects in the images.

4. Deep Learning Model

This experiment aims to showcase the results of the most impactful preprocessing techniques. We excluded the results of some preprocessing methods mentioned above because they gave the same results as the base model. The focus was on exploring the ones that provided significantly different results.

4.1 Data and System Preparation

The data preparation process is started by loading the base images for the base model. Subsequently, we applied transformations separately to the data and fed it into the model (Deepa et al., 2023). Additionally, it utilized some additional functions from PyTorch transformers for further preprocessing techniques that could enhance our results. The system was configured with CUDA for faster training times than a standard CPU.

4.2 Model Implementation

To implement our deep learning model, we employed PyTorch to expedite the development process and CUDA to accelerate training. We opted against creating a custom model from scratch instead of utilizing RESNET50. RESNET, a residual learning framework, facilitates the training of networks significantly more profoundly than those employed in prior models (He et al., 2015). Initially designed for classifying the ImageNet dataset, which consists of 1000 different classes, we tailored the model to classify the 10 classes of the CIFAR10 dataset. Rather than training RESNET50 weights from the ground up, we aimed to leverage the knowledge gained from the thousands of images in the ImageNet dataset through transfer learning (TL). Our approach involved loading a pre-trained RESNET50 model with weights and adjusting the last layer of the neural network (output layer) to accommodate the 10 classes of the CIFAR10 dataset instead of the original 1000, as shown in Figure 2.

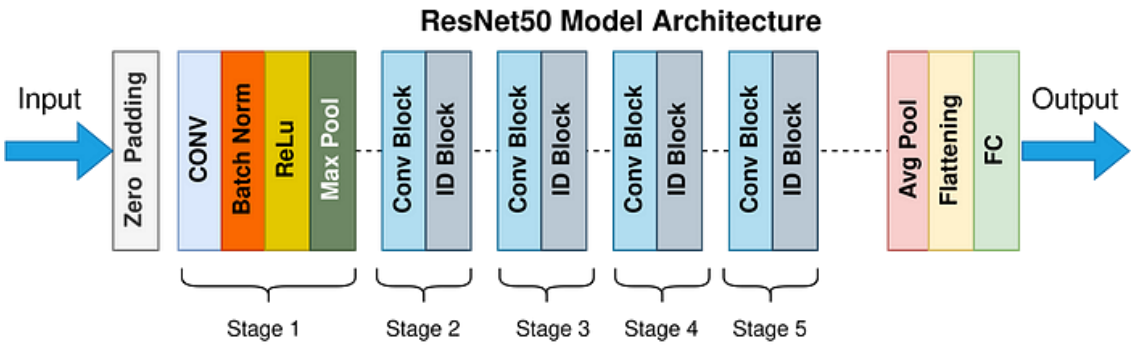

Figure 2: *ResNet50 – Gorlapraveen123*

In summary, our model was built using the RESNET architecture with pre-trained weights (Singh et al., 2023). We made a specific modification by adjusting the fully connected (FC) layer in the architecture to output 10 classes instead of the original 1000. The decision to use RESNET and the rationale behind that choice will not be discussed further, as our focus here is to provide an overview of how we formulated our model for training on the CIFAR10 dataset.

4.3 Training Process

Before training, we observed the most impactful preprocessing technique, rather than the one that produces the best model. The aim was to understand how preprocessing can significantly influence learning, whether positively or negatively. To comprehend why preprocessing had a substantial impact on our results, let's briefly delve into how deep learning models operate. Deep learning models utilize stochastic gradient descent (SGD) to determine the model

parameters that best fit the relationship between truth and prediction (Haji & Abdulazeez, 2021). This optimization process relies on derivatives to determine whether to increase or decrease the parameters, ultimately guiding the model toward the global minimum. However, there is a possibility of the model getting stuck in a local minimum, requiring the learning process to be reiterated to find a new starting point in the hopes of reaching a lower minimum and eventually converging (Yousif & AlRababaa, 2013). The data is prepared and fed to the proposed network with unprocessed images. Subsequently, we introduced various images after applying different preprocessing techniques.

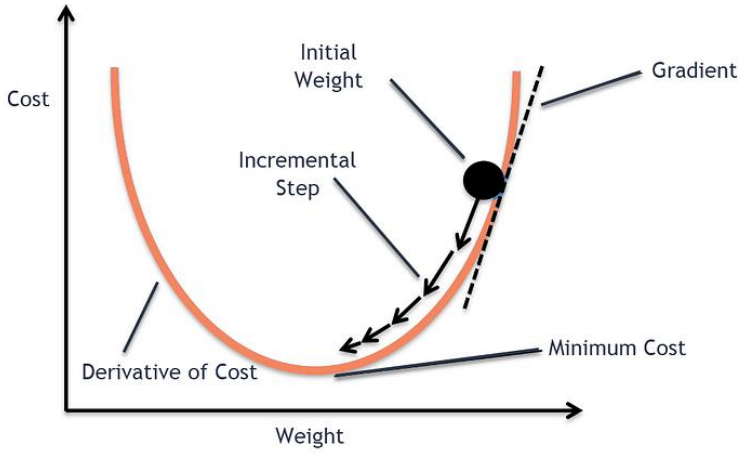

Figure 3: *stochastic gradient descent (Huang et al., 2023)*

4.4 Testing Process

It's crucial to use a distinct set of images for testing than the one used for training. This practice is imperative in machine learning to avoid biased results and prevent a misrepresentation of the model's accuracy. To address this, we partitioned our images into a 50,000-image training set and separated 10,000-image validation and testing sets. To ensure consistency, the exact same data was used for evaluating the performance of different preprocessing methods, minimizing variations in the datasets. The obtained results were interesting, revealing differences between unprocessed and processed images. The results show that the model accuracy is improved with the following preprocessing methods:

- Normalization and Histogram Equalization
- Random Cropping & Rotations
- All (combined) 93%

The results achieved after 20 epochs are summarized in Table 2.

Table 2: A summary of results achieved

Preprocessing Method	Testing Accuracy
None	85%
Normalization & Histogram Equalization	91%
Random Cropping & Rotation	89%
ALL	93%

There is a significant jump in accuracy with preprocessing. After obtaining these results, the study shifted focus to understanding the reason behind the improved accuracy, as shown in Figure 4. Initially, this study believed preprocessing directly enhanced our model's accuracy. However, while the results indicate improvement, the enhanced performance is not solely due to preprocessing; instead, it is attributed to a different factor related to our optimization method—stochastic gradient descent. To gain deeper insights, we employed Wandb, a popular tool among machine learning developers, to graph various parameters and results, providing a more intuitive visualization. Examining the training accuracy graph as shown in Figure 4, we can observe that all models are moving to convergence; even the one fed with unprocessed data seems like it will converge if we give it more training iterations.

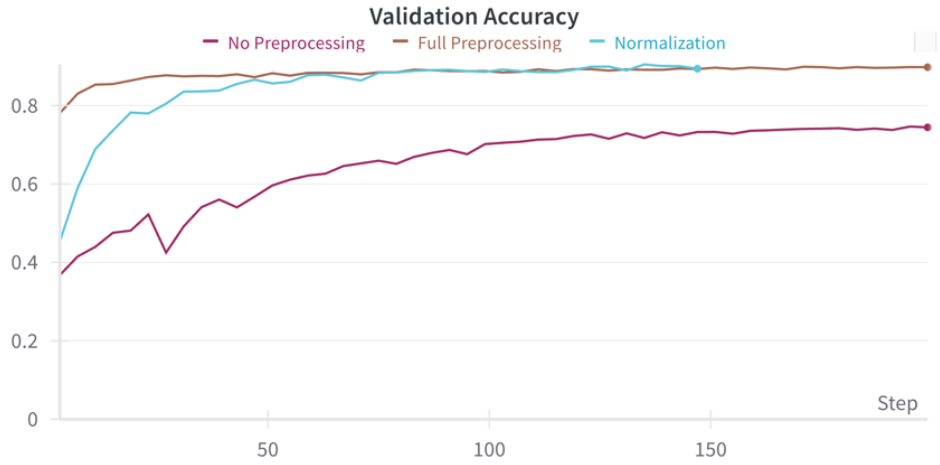

Figure 4: The results of model validation accuracy

This experiment likely involves training a machine learning model. By increasing the number of epochs to 120, the model is given more time to learn from the data and improve its accuracy. The adjustments may have included changes to the model architecture, hyperparameters, or input data. Figure 6 represents the model's performance, such as a graph showing how its accuracy or loss changes over time. Eventually, all models would converge, as experimentation is aimed. The model trained on preprocessed data took 20 epochs, approximately 15 minutes, to converge, while the one trained on unprocessed data took about 90 epochs, approximately an hour. This represents a

significant time difference for a relatively small dataset. The difference would be even more pronounced if a larger dataset was included. As a result, the study provides reasoning for the following preprocessing methods: normalization, random rotation, and cropping.

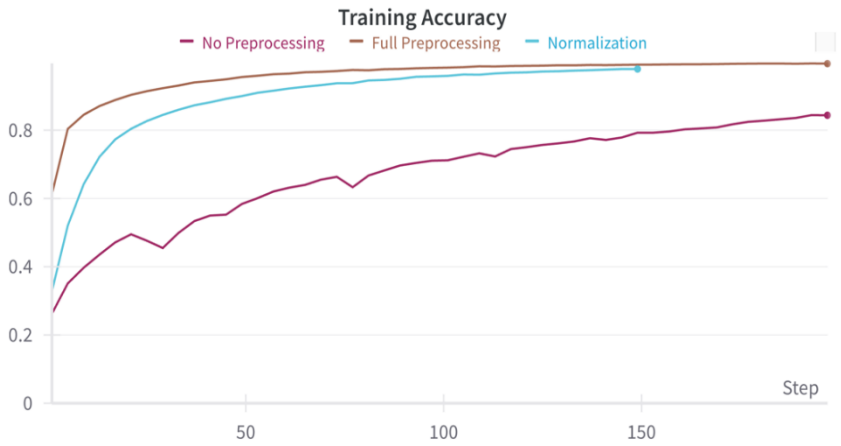

Figure 5: The results of model training accuracy

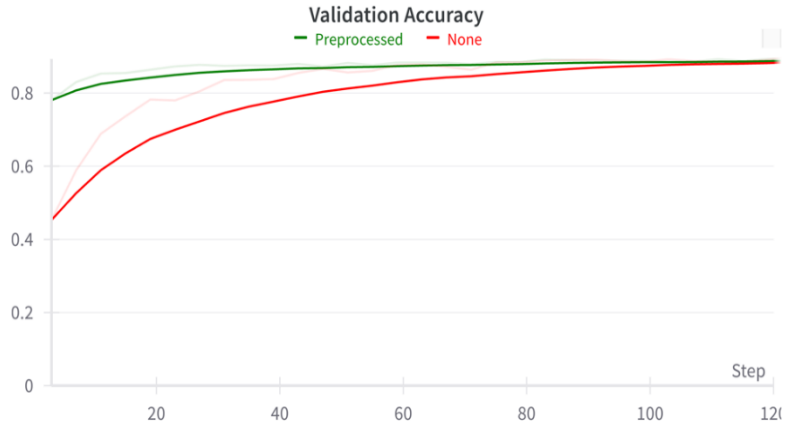

Figure 6: The results comparison of model validation accuracy

4.5 Normalization

The goal of normalization is to scale the input data to be on a similar scale. This helps the gradient descent converge faster (Yousif & Kazem, 2021). If we were to visualize this, it would give the error surface a more circular or spherical shape as shown in Figure 7. This reduction in curvature minimizes unnecessary steps during gradient descent in the optimal direction. With fewer curvatures, gradient descent moves toward the global minimum more efficiently (Wang et al, 2022). It goes directly to the minimum, making the learning process easier for the model.

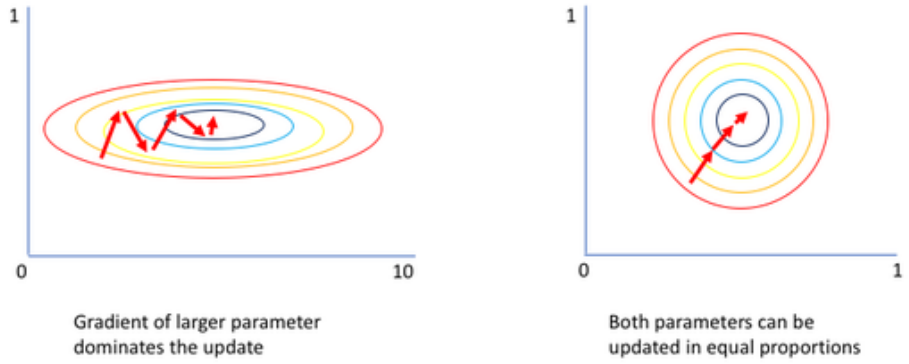

Figure 7: The results comparison of model validation accuracy (Wang et al, 2022)

4.6 Random Cropping and Rotation

There is no concrete reasoning for the improved accuracy when performing random cropping and rotation. The only theory we have is that our model sees a different form of the same image in every iteration, possibly reducing overfitting. Overfitting tends to result in very high training accuracy but low testing accuracy because the model is memorizing the input (Fekihal & Yousif, 2012). By presenting the model with different forms of the same input through random cropping and rotation, it may prevent overfitting and optimize its parameters more efficiently. This, in turn, leads to better testing accuracy.

5. Conclusion

This The experimentation with preprocessing techniques on a deep learning model, specifically using RESNET50, provided notable insights. It observed a significant improvement in model accuracy with specific preprocessing methods, such as normalization and random cropping with rotation. This is because preprocessing speeds up the training process and saves computation time and resources. Both models converged, but the preprocessed one took 20 epochs (~15 mins), while the unprocessed one took 90 epochs (~1 hour). This experiment likely involves training a machine learning model. By increasing the number of epochs to 120, the model is given more time to learn from the data and improve its accuracy. The adjustments may have included changes to the model architecture, hyperparameters, or input data. Eventually, all models would converge, as experimentation is aimed. There is a significant time difference for a relatively small dataset. The difference would be even more pronounced if a more extensive dataset were included. As a result, the study provides reasoning for implementing different preprocessing methods, such as normalization, random rotation, and cropping. This study on preprocessing techniques with

RESNET50 yielded positive results and insights. However, certain limitations and considerations exist to acknowledge, such as dataset specificity, task dependency, overfitting concerns, hyperparameter sensitivity, trade-off analysis, comparison with baseline models, computational resources consideration, generalization to other architectures, and reproducibility and transparency. The study highlighted that specific preprocessing techniques can enhance model accuracy on a small dataset. However, it is essential to note that the effectiveness of these techniques may differ across various datasets. Therefore, evaluating whether the observed improvements apply to larger and more diverse datasets is essential. Also, the significant decrease in training time observed with preprocessing might lead to concerns regarding overfitting, especially when working with a smaller dataset. Further analysis, such as validation on an independent dataset, is necessary to ensure that the improvements are only partially due to overfitting.

Acknowledgment: The research leading to these results has received no Research Grant Funding.

Author contribution: All authors have contributed, read, and agreed to the published version of the manuscript results.

Conflict of interest: The authors declare no conflict of interest.

References

[1]. Al-Hatmi, M. O., & Yousif, J. H. (2017). A review of Image Enhancement Systems and a case study of Salt &pepper noise removing. International Journal of Computation and Applied Sciences (IJOCAAS), 2(3), 171-176.
[2]. Alighaleh, P., Khosravi, H., Rohani, A., Saeidirad, M. H., & Einafshar, S. (2022). The detection of saffron adulterants using a deep neural network approach based on RGB images taken under uncontrolled conditions. Expert Systems with Applications, 198, 116890.
[3]. Alkishri, W., Widyarto, S., Yousif, J. H., & Al-Bahri, M. (2023). Fake Face Detection Based on Colour Textual Analysis Using Deep Convolutional Neural Network. Journal of Internet Services and Information Security (JISIS), 13(3), DOI: 10.58346/JISIS.2023.I3.009.
[4]. Atomi, W. H. (2012). The effect of data preprocessing on the performance of artificial neural networks techniques for classification problems (Doctoral dissertation, Universiti Tun Hussein Onn Malaysia).
[5]. Calderon, S., Fallas, F., Zumbado, M., Tyrrell, P. N., Stark, H., Emersic, Z., ... & Solis, M. (2018, October). Assessing the impact of the deceived non local means filter as a preprocessing stage in a convolutional neural network based approach for age estimation using digital hand x-ray images. In 2018 25th IEEE International Conference on Image Processing (ICIP) (pp. 1752-1756). IEEE.
[6]. Deepa, T., Satyavathy, G., Priya, A., & Hamrish, S. S. (2023). Deep Learning with Pytorch: Siamese Network. Tuijin Jishu/Journal of Propulsion Technology, 44(4), 4412-4421.
[7]. Fekihal, M. A., & Yousif, J. H. (2012). Self-organizing map approach for identifying mental disorders. International Journal of Computer Applications, 45(7), 25-30.
[8]. Ghandour, C., El-Shafai, W., & El-Rabaie, S. (2023). Medical image enhancement algorithms using deep learning-based convolutional neural network. Journal of Optics, 1-11.
[9]. Haji, S. H., & Abdulazeez, A. M. (2021). Comparison of optimization techniques based on gradient descent algorithm: A review. PalArch's Journal of Archaeology of Egypt/Egyptology, 18(4), 2715-2743.
[10]. He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. ArXiv. /abs/1512.03385.
[11]. Hasoon, F. N., Yousif, J. H., Hasson, N. N., & Ramli, A. R. (2011). Image enhancement using nonlinear filtering based neural network. Journal of Computing, 3(5), 171-176.

[12]. Huang, L., Qin, J., Zhou, Y., Zhu, F., Liu, L., & Shao, L. (2023). Normalization techniques in training dnns: Methodology, analysis and application. IEEE Transactions on Pattern Analysis and Machine Intelligence.
[13]. Khamis, Y., & Yousif, J. H. (2022). Deep learning Feedforward Neural Network in predicting model of Environmental risk factors in the Sohar region. Artificial Intelligence & Robotics Development Journal, 201-2013.
[14]. Lapid, R., & Sipper, M. (2022, July). Evolution of activation functions for deep learning-based image classification. In Proceedings of the genetic and evolutionary computation conference companion (pp. 2113-2121).
[15]. Öztürk, Ş., & Akdemir, B. (2018). Effects of histopathological image pre-processing on convolutional neural networks. Procedia computer science, 132, 396-403.
[16]. Pajankar, A., & Joshi, A. (2022). Hands-on Machine Learning with Python: Implement Neural Network Solutions with Scikit-learn and PyTorch. Apress.
[17]. Şengöz, N., Yiğit, T., Özmen, Ö., & Isik, A. H. (2022). Importance of preprocessing in histopathology image classification using deep convolutional neural network. Advances in Artificial Intelligence Research, 2(1), 1-6.
[18]. Singh, V. K., Sharma, K., & Sur, S. N. (2023). A survey on preprocessing and classification techniques for acoustic scene. Expert Systems with Applications, 120520.
[19]. Tabik, S., Peralta, D., Herrera-Poyatos, A., & Herrera Triguero, F. (2017). A snapshot of image pre-processing for convolutional neural networks: case study of MNIST.
[20]. Wang, Y., He, Y., & Zhu, Z. (2022). Study on fast speed fractional order gradient descent method and its application in neural networks. Neurocomputing, 489, 366-376.
[21]. Yousif, J. H., & Kazem, H. A. (2021). Prediction and evaluation of photovoltaic-thermal energy systems production using artificial neural network and experimental dataset. Case Studies in Thermal Engineering, 27, 101297.
[22]. Yogeshwari, M., & Thailambal, G. (2023). Automatic feature extraction and detection of plant leaf disease using GLCM features and convolutional neural networks. Materials Today: Proceedings, 81, 530-536.
[23]. Yousif, J. H., & AlRababaa, M. S. (2013). Neural technique for predicting traffic accidents in Jordan. Journal of American Science, 9(11), 528-525.
[24]. Zhou, K., Oh, S. K., Pedrycz, W., & Qiu, J. (2023). Data preprocessing strategy in constructing convolutional neural network classifier based on constrained particle swarm optimization with fuzzy penalty function. Engineering Applications of Artificial Intelligence, 117, 105580.