Comparative Analysis of Image Classification using DeepCNN Models on MNIST

Gargeya Sharma
MSc. Artificial Intelligence, EECS
Queen Mary University of London
London, United Kingdom
ec22146@qmul.ac.uk

***Abstract*—This paper conducts an exhaustive comparative analysis of the implications of using raw and mean-reduced MNIST datasets on MobileNetV2 architecture for image classification. The obtained results from this analysis drive the further investigation of deep image classification on mean-reduced MNIST dataset using three prominent Convolutional Neural Network architectures: MobileNetV2, ResNet18, and GoogLeNet. Through extensive experiments, this paper delineates the performance variations, accuracy rates, and trade-offs associated with each architecture. Training is performed from scratch using the PyTorch framework where evaluation methodology relies on training logs such as loss and accuracy plots, training time stored in TensorBoard, and corresponding class-wise performance evaluation using confusion matrices. My findings provide a critical analysis of the impacts of various deep-learning models on MNIST dataset, paving the way for more efficient and accurate model selection in the field of image classification.**

***Index Terms*—image classification, deep convolutional neural networks, comparative analysis, experiments, MNIST dataset**

I. Introduction

A strong rise of AI shifted the paradigm in the study of computer science as a result of the development of Machine Learning (ML) and deep learning (DL). In particular, Convolutional Neural Networks (CNNs)(Y. Lecun et al., 1998) [1], a subset of deep learning models, have excelled in image recognition, classification and other vision-related tasks. However, a number of variables, such as the architecture chosen, the dataset preprocessing methods used, learning rates, the number of epochs, and batch size affect how well these models perform. This study aims to unravel the interplay between these factors, specifically focusing on three CNN architectures: MobileNetV2 (Sandler et al., 2019) [2], ResNet18 (He et al., 2015) [3], and GoogLeNet (Szegedy et al., 2014) [4], and two preprocessing methods: raw and mean-reduced data [1] [5] [6] on MNIST handwritten digit dataset (LeCun,Y. et al., 2010) [7]. The objective is to provide an empirical foundation for choosing the right combination of the model architecture and preprocessing technique for optimal performance and validate if these findings are robustly applicable to other datasets like CIFAR10 (Krizhevsky, A. et al., 2009) [8] or not.

A. Abbreviations and Acronyms

Defining all the abbreviations and Acronyms used in the paper. Artificial Intelligence (AI), Machine Learning (ML), Deep Learning (DL), Modified National Institute of Standards and Technology (MNIST), Convolutional Neural Networks (CNN), Canadian Institute For Advanced Research (CIFAR), Residual Networks (ResNet), Learning rate (Lr)

II. Datasets

A. MNIST Dataset

The main focus of the paper is the MNIST dataset [7]. The MNIST database is an extensive collection of handwritten digits, widely used for training and testing in the field of machine learning. It contains 60,000 training images and 10,000 testing images, each of 28x28 pixel dimensions. The raw dataset represents the original images, whereas the mean-reduced dataset is the result of subtracting the mean of the image pixel values from each pixel, a common normalization technique. This study explores the implications of using both types of data on the performance of the MobileNetV2 architecture where further inter-model performance comparison is done on the mean-reduced version of the dataset.

B. CIFAR-10 Dataset

The CIFAR-10 dataset [8] is used as a supporting dataset in this paper as it is a widely recognized benchmark in the field of computer vision and image classification. CIFAR-10 consists of 60,000 32x32 colour images categorized into ten distinct classes, each representing common objects such as airplanes, automobiles, birds, cats, and more. The dataset provides a challenging and diverse set of images, making it an ideal choice for evaluating the performance and generalization ability of image classification algorithms. Its relatively small image size facilitates efficient training and testing, while still capturing essential visual features. With its well-defined class labels and balanced distribution, CIFAR-10 enables me to analyze the impact of different architectures used with MNIST and briefly validate my findings.

III. Model Architectures

MobileNetV2 [2]: Introduced by researchers at Google, MobileNetV2 represents a significant evolution over its predecessor, MobileNetV1 (Howard et al., 2017) [9]. Its primary innovation lies in the introduction of an inverted residual structure with linear bottlenecks. Unlike traditional residual models, MobileNetV2 introduces shortcuts between the

lightweight depthwise convolutions, as opposed to the computationally heavy projection layers. This results in a substantial reduction in computational complexity, facilitating the deployment of sophisticated machine vision capabilities on resource-constrained mobile devices. Moreover, the architecture is designed to provide an optimal balance between computational cost and model performance, making it a versatile choice for various applications.

ResNet18 [3]: As a member of the ResNet family, ResNet18 embodies a breakthrough in the design of deep learning architectures. Comprising 18 layers, this model effectively addresses the notorious vanishing gradient problem through the use of residual blocks or "skip connections" (Orhan & Pitkow, 2018) [10]. These allow the model to learn identity functions, ensuring that the addition of more layers contributes positively to performance or, at the very least, does not degrade it. This design principle enables the training of considerably deeper networks (Mallat, 2016) [11] (Huang et al., 2018) [12] than was previously possible, thus enhancing their ability to learn complex patterns in the data. ResNet18, therefore, remains a strong contender for tasks requiring advanced feature extraction capabilities.

GoogLeNet [4]: Also known as Inception-v1, GoogLeNet introduced a novel concept called the inception module. This module comprises parallel operations with different receptive field sizes, enabling the network to adaptively capture information at various scales. The inception module also includes dimensionality reduction via 1x1 convolutions before the expensive 3x3 and 5x5 convolutions, significantly reducing the computational cost. This allows for an increase in network depth and width without an associated exponential increase in computational resources. As a result, GoogLeNet is able to deliver impressive performance on image classification tasks without imposing prohibitively high computational demands.

The aim of this paper is to conduct a comprehensive performance analysis of these architectures using the MNIST dataset. By evaluating the strengths and weaknesses of each architecture in relation to one another, we hope to provide practical insights that can guide the selection of appropriate architectures for different tasks and operational environments.

IV. Experimentation

A. Data Preprocessing: An Evaluation of Raw and Mean-Reduced MNIST Dataset on MobileNetV2

In the field of deep learning, data preprocessing plays a crucial role in model performance and efficiency. One common preprocessing technique is the mean reduction or mean normalization, which involves subtracting the mean of the dataset from each data point. This process results in a dataset with a zero mean, effectively centering the data. This technique is particularly beneficial as it aids in smoother optimization of the loss function, thereby facilitating more efficient learning (Y. Lecun et al., 1998) [1].

In the present study, a comparative analysis was performed using MobileNetV2 with two different settings of the MNIST dataset: raw and mean-reduced. While the ultimate accuracy achieved by the model was not significantly different between the two settings, the study observed noticeable differences in the optimization process. With the RAW dataset, every time when the training was performed from scratch, the optimization looked inconsistent and rough (relatively sharper zig-zag optimization). Even the classes that were misclassified (as seen from the confusion matrix) were also inconsistent with each individual training. All in all, poor consistency was observed with respect to what is being learned, and as a result, we can say that the multidimensional loss plane of the raw data distribution is more complex than the variance loss plane we get from subtracting the mean from the raw dataset.

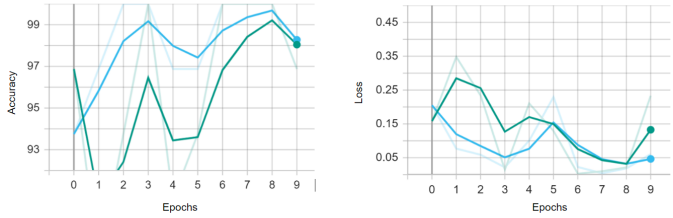

Fig. 1. Nature of Training with raw V/s mean-reduced MNIST
(Left figure: Accuracy plot, Right figure: Loss Plot (light-blue: mean-reduced dataset, green: Raw dataset))

As shown in Fig.1, When trained on the mean-reduced dataset, MobileNetV2 exhibited a smoother learning curve compared to its performance with the raw dataset. This outcome aligns with the understanding that mean reduction helps in better conditioning of the optimization problem, preventing extreme variations in the loss function, and thus allowing the optimization algorithm to converge faster (Goodfellow, Bengio, & Courville, 2016) [13].

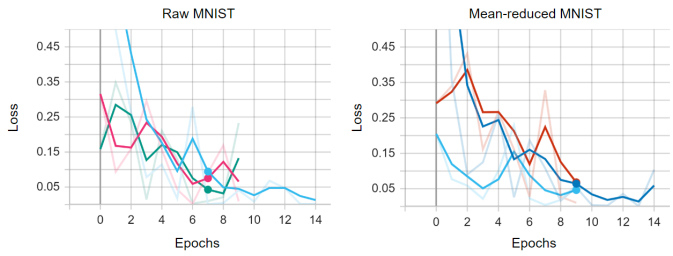

Fig. 2. Loss Curve of MobileNetV2 with different learning rates during training with raw V/s mean-reduced MNIST
(Left figure: (Light blue: Lr = 0.0001 epoch = 15, Magenta: Lr = 0.0002 epoch = 10, Green: Lr = 0.0004 epoch = 10), Right figure: (Dark blue: Lr = 0.0001 epoch = 15, Red: Lr = 0.0002 epoch = 10, Light Blue: Lr = 0.0004 epoch = 10))

With the help of Fig. 2 where several MobileNetV2 models are trained with different learning rates, we can validate the findings that a mean-reduced dataset leads to smoother optimization but despite that, the figure underscores the benefits of mean-reduction preprocessing, particularly in terms of loss reduction as the final model accuracy on the mean-reduced dataset was not significantly different from the raw

TABLE I
PERFORMANCE COMPARISON OF MOBILENETV2 ON RAW V/S MEAN-REDUCED MNIST

	MobileNetV2 (Lr=0.0004) and Smoothing with Tensorboard			
	MobileNetV2 (Raw MNIST)	MobileNetV2 (Raw MNIST) (Smoothing factor = 0.5)	MobileNetV2 (Mean-reduced MNIST)	MobileNetV2 (Mean-reduced MNIST) (Smoothing factor = 0.5)
Training Accuracy (%)	96.88	98.04	96.88	98.28
Testing Accuracy (%)	98.71	98.71	98.68	98.68
Loss	0.233	0.132	0.06	0.046
Training Time (seconds)	379	379	427	427

data. This indicates that while mean reduction may aid in optimization, it does not necessarily translate to significantly improved performance in terms of final accuracy. Hence, the choice of data preprocessing method may depend on the specific requirements of a project, such as time constraints or computational resources.

Table 1 presents a detailed comparison of the MobileNetV2 model trained under different conditions: using raw MNIST dataset versus mean-reduced MNIST dataset, each with and without a TensorBoard [15] smoothing factor of 0.5 applied post-training. The evaluated metrics include training and testing accuracy, loss, and training time. The following discussion provides an analysis of these results.

Training and Testing Accuracy: The training accuracy across the four configurations displays slight variations. Without smoothing, the training accuracy remains the same (96.88%) for both the raw and mean-reduced dataset. This suggests that mean-reduction preprocessing does not significantly impact the learning performance of MobileNetV2 in terms of training accuracy.

When smoothing is applied, both configurations show an increase in training accuracy, with the mean-reduced dataset achieving a slightly higher accuracy (98.28%) compared to the raw dataset (98.04%). The increase seen after smoothing is due to the TensorBoard's effect of averaging out fluctuations in the learning curve, presenting a more stable view of the model's performance over time.

Interestingly, the testing accuracy remains unchanged when switching from the raw to the mean-reduced dataset, both with and without smoothing. This suggests that the mean-reduction preprocessing and smoothing do not materially affect the model's ability to generalize to unseen data.

Loss: The loss values indicate a notable advantage of mean-reduction preprocessing, with a decrease from 0.233 (raw) to 0.06 (mean-reduced) without smoothing, and from 0.132 (raw) to 0.046 (mean-reduced) with smoothing. This suggests that mean-reduction preprocessing aids in a smoother optimization process and potentially a better fit to the training data.

Training Time: The training time is slightly longer for the mean-reduced dataset (427 seconds) compared to the raw dataset (379 seconds), both with and without smoothing. The increase in training time could be attributed to the additional computational operations required for mean-reduction preprocessing.

In summary, the improvements in training accuracy appear modest, and the testing accuracy remains largely unaffected. Smoothing, on the other hand, provides a more stable view of the model's performance over time, aiding in the interpretation of the learning process without altering the actual model performance. These findings highlight the nuanced effects of preprocessing techniques and post-training smoothing on model performance and the importance of a detailed understanding of these effects when training and evaluating deep learning models.

B. Comparison of Deep-CNN Models in Classifying MNIST Images

As discussed in the previous section, if not with loss, training or testing accuracy, the mean-reduction of a dataset does help with the smoother training process in terms of optimization. Hence, from this point onwards, this paper will use the mean-reduced MNIST dataset to train the models and compare their performances with respect to trade-offs that arise with the user's needs and problem statements. These models have been trained from scratch on the MNIST dataset to be eligible for comparison. Fig.3 shows the run-time screenshots of these models in the form of training loss and accuracy plots.

Fig.3. clearly shows that ResNet18 and GoogLeNet took their performance near 100% on the training dataset which is 1% better if not more than what MobileNetV2 could do with several training attempts on different learning rates. The right plot that shows the loss is much similar to the accuracy plot except that GoogLeNet and MobileNetV2's loss is closer to each other than ResNet18 and GoogLeNet. This disparity in their behaviour needs to be checked by comparing the training accuracies with testing accuracies and see which model shows signs of higher variance between the two and is more prone to overfitting. This issue will be covered later using the training and testing results from Table 2.

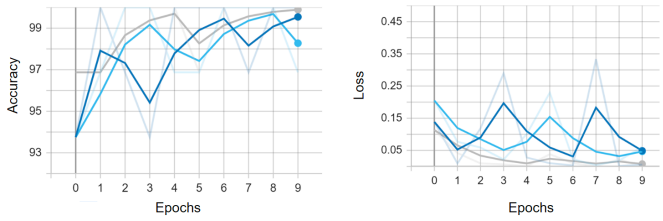

Fig. 3. Nature of Training with raw V/s mean-reduced MNIST
(Left figure: Accuracy plot, Right figure: Loss Plot (light-blue: mean-reduced dataset, green: Raw dataset))

Table 2 presents a comparative analysis of three different model configurations, including MobileNetV2, ResNet18, and GoogLeNet, each with and without a TensorBoard smoothing factor of 0.5 applied to the displayed results. The metrics considered include training and testing accuracy, loss, training time, and the number of learnable parameters. The results and their implications are discussed below.

Training and Testing Accuracy: The training accuracies of all the models are relatively high, with ResNet18 and GoogLeNet achieving perfect accuracy (on the last epoch). MobileNetV2 demonstrates a training accuracy of 96.88% without smoothing and 98.28% with smoothing. The slight increase in the smoothed results is due to the effect of the smoothing factor, which averages out variations in the learning curve and can provide a more stable view of the model's performance over time. However, it's crucial to note that the underlying model performance is not affected by this smoothing process; it merely presents the data differently.

The testing accuracies for all models are very close to their respective training accuracies, indicating good generalization and minimal overfitting. The smoothing factor does not alter these values, suggesting a consistent performance across epochs.

Loss: The loss values vary across the models, with ResNet18 showing the lowest loss without smoothing (0.00018), indicating a superior fit to the training data. Applying a smoothing factor results in a more uniform view of the loss values across epochs, making the learning process appear more stable. However, the fundamental model's loss performance remains unchanged by this representation.

Training Time: The training time, unaffected by TensorBoard smoothing, reflects the computational efficiency of the different architectures. GoogLeNet has the longest training time (427 seconds), while ResNet18 is the fastest (267 seconds), and MobileNetV2 falls in the middle (533 seconds). These times are indicative of the architectural complexity and the learning procedure in each model.

Number of Learnable Parameters: The number of learnable parameters is a fundamental characteristic of each model, unrelated to TensorBoard smoothing. ResNet18 has the highest number of parameters (11 million), followed by GoogLeNet (6.8 million), and MobileNetV2 (3.4 million). This metric impacts both the model's capacity to learn complex patterns and the computational resources required for training.

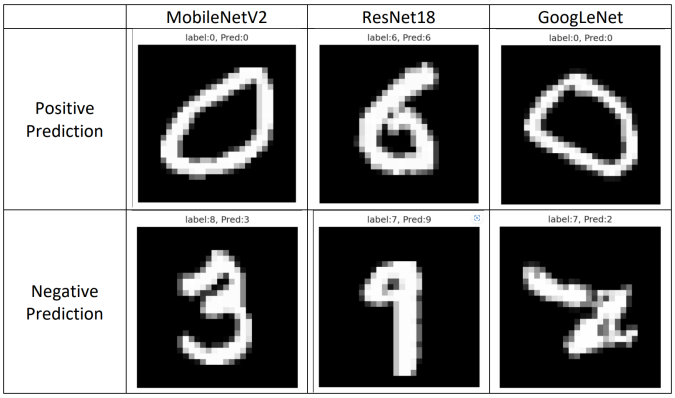

Fig. 4. Positive and Negative Predictions by MobileNetV2, ResNet18 and GoogLeNet over test dataset

As depicted in Fig. 4, the negative predictions per model underscore the presence of potentially mislabelled samples, likely attributable to human error during the dataset's creation. This error introduces ambiguity in feature maps, complicating the model's training process and possibly impeding the attainment of optimal accuracy during testing. Even when the model attains a robust representation of the classes, the underlying quality of the data can serve as a limiting factor to its performance. This phenomenon aligns with the concept of Bayes error (Antos et al., 1999) [16], which acknowledges an irreducible error rate even for an ideal model due to inherent noise or uncertainties within the data. It is a compelling reminder that while our models can learn increasingly sophisticated representations, their performance remains fundamentally tied to the quality of the data they are trained on.

V. Analysis and Observations

Upon careful observation of the results presented in Figure 3, Table 2, and Figure 5, it can be discerned that ResNet18 outperforms the other two architectures, followed by GoogLeNet and MobileNetV2. However, it's important to note that these models exhibit different trade-offs in terms of the number of parameters and layers they incorporate.

ResNet18, despite showing superior performance, utilizes around 11 million parameters, resulting in increased memory consumption and model size. GoogLeNet, on the other hand, exhibits competitive performance, almost on par with ResNet18, while utilizing only 6.8 million parameters - more than a 35% reduction compared to ResNet18.

As shown in Figure 5, the confusion matrices from our models during the testing time provide insight into the distinct representations of information learned during their respective training phases. The unique architectural design, internal feature representations, and training procedures inherent in each model collectively contribute to a diverse range of error rates per class in the dataset. Consequently, these models exhibit distinct levels of accuracy, precision, and recall for each class, underscoring the significance of model selection based on the specific requirements of the classification task at hand.

TABLE II
PERFORMANCE COMPARISON BETWEEN MOBILENETV2, RESNET18 AND GOOGLENET

	MobileNetV2	MobileNetV2 (smoothing factor= 0.5)	ResNet18	ResNet18 (smoothing factor= 0.5)	GoogLeNet	GoogLeNet (smoothing factor = 0.5)
Training Accuracy (%)	96.88	98.28	100	99.89	100	99.54
Testing Accuracy (%)	98.68	98.68	99.06	99.06	99.11	99.11
Loss	0.06	0.046	0.00018	0.0078	0.006	0.049
Training Time (seconds)	427	427	267	267	533	533
No. of Learnable Parameters (Millions)	3.4	3.4	11	11	6.8	6.8

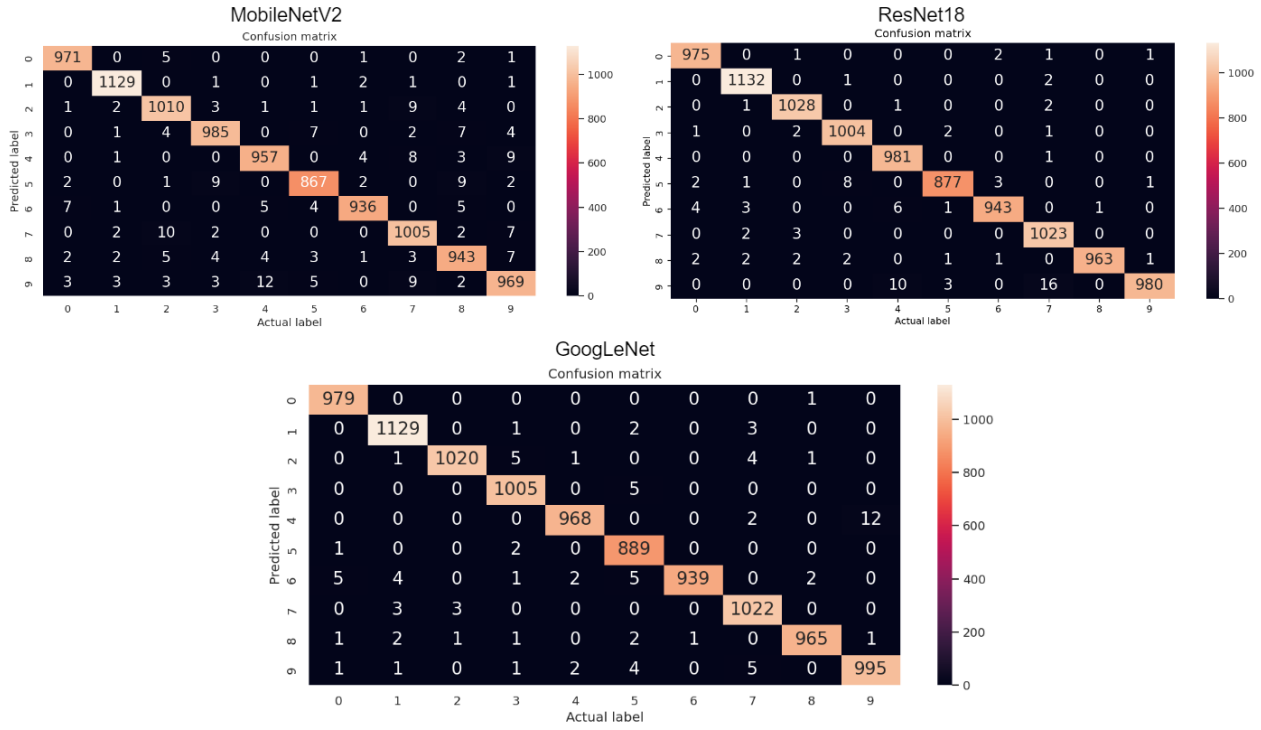

Fig. 5. Confusion Matrices of MobileNetV2, ResNet18 and GoogLeNet over test dataset

A keen examination of Table 2 reveals a greater variance between training and testing accuracy for ResNet18 compared to the other two models. This observation suggests a potential overfitting scenario with ResNet18, necessitating further investigation on a different dataset such as CIFAR-10 to ascertain whether this pattern persists.

ResNet18 demonstrates superior convergence speed, achieving the highest accuracy in the quickest time amongst the three models. However, it also shows the highest variance and houses the most parameters. This observation serves to reiterate that no model uniformly outperforms the others across all tasks and applications. The choice of model will inevitably involve trade-offs between various characteristics, such as model size, training time, and performance.

In a scenario where rapid training and high accuracy are paramount, ResNet18 would be the preferred choice. However, to avoid overfitting, an early stopping strategy, a popular form of regularization (Santos & Papa, 2022) [14], could be employed. This would halt the training process as soon as the model's learning plateaus, thus striking a balance between training speed and generalization.

Conversely, if the requirement emphasizes a compact model over high accuracy, MobileNetV2 would be the optimal choice. With only 3.8 million parameters, MobileNetV2 offers

TABLE III
MOBILENETV2, RESNET18, GOOGLENET ON CIFAR-10 WITH POST-SMOOTHING OF 0.5 IN TENSORBOARD

	MobileNetV2	ResNet18	GoogLeNet
Train Accuracy (%)	79.83	85.03	69.24
Test Accuracy (%)	69.63	75.63	72.95
Loss	0.6121	0.3404	0.9111

commendable performance while maintaining a model size significantly smaller than that of ResNet18 and GoogLeNet. This compactness, coupled with competitive accuracy, underscores MobileNetV2's suitability for scenarios where model size is a critical constraint.

In conclusion, the selection of an appropriate deep learning model is contingent upon the specific requirements and constraints of the task at hand, and a thorough understanding of the trade-offs involved is essential for making an informed decision.

Further empirical evidence to support our comparative analysis and evaluate some of the doubts was gathered by conducting image classification with MobileNetV2, ResNet18, and GoogLeNet on the CIFAR-10 dataset under slightly varied hyperparameters. The batch size was altered from 64 to 32, and optimal learning rates were employed for each model to enable rapid and comparable model training.

From the smoothened results shown in Table 3, a discernible difference of approximately 10% is observed between training and testing accuracies for MobileNetV2 and ResNet18. This disparity is significantly more pronounced in ResNet18, which, despite achieving a lower loss value (0.3404) compared to MobileNetV2 (0.6121) and GoogLeNet (0.9111), does not exhibit proportionally superior testing accuracy. ResNet18 achieves a training accuracy of 85.03% and a testing accuracy of 75.63%. In comparison, GoogLeNet, which boasts a closer correlation between the two, shows a noticeably lower training accuracy of 69.24%, than testing accuracy of 72.95%.

These observations underscore a higher propensity for overfitting in ResNet18 as well as MobileNetV2, as indicated by the larger gap between its training and testing accuracies. Interestingly, despite this overfitting tendency, ResNet18's loss is markedly lower at 0.3404, compared to MobileNetV2's loss of 0.6121 and GoogLeNet's loss of 0.9111. This suggests that ResNet18 is more efficient at minimizing loss within the training dataset, but this efficiency does not fully translate to improved performance on the testing data, which makes it the strongest candidate among the three to suffer from overfitting. This challenges the model's ability to generalize, which is vital for real-world applicability where the model will encounter diverse and previously unseen data.

To counteract this, it may be prudent to introduce more rigorous regularization techniques [14] like early stopping as mentioned above, or an even better option would be stronger data augmentation [14] before training as a form of regularization when using ResNet18 to reduce overfitting and enhance its generalization performance on the dataset.

VI. CONCLUSION

In the course of this research, we have embarked on an extensive exploration of the performance of three prominent convolutional neural networks: MobileNetV2, ResNet18, and GoogLeNet. We have scrutinized their performance on image classification tasks, using both raw and mean-reduced MNIST datasets, and made significant observations that influence our understanding of these models and their optimal utilization.

Our findings underscore the significance of preprocessing techniques in the performance of these models. Particularly, the utilization of mean-reduced data, while not significantly altering the final optimization outcomes, introduces a smoother and more efficient training process. This technique, therefore, offers the potential for enhancing model efficiency and reducing optimization noise, thereby promoting a quicker convergence to the model's optimal state.

Upon further analysis using the mean-reduced MNIST dataset, we observed distinct performance characteristics for each model. ResNet18 exhibited superior performance in terms of training accuracy and speed, despite its larger model size. However, its high tendency for overfitting necessitates caution, as this could impact its generalization capabilities on unseen data.

GoogleNet, on the other hand, demonstrated robust performance, with high accuracies and significantly fewer parameters than ResNet18. However, it requires a longer training duration to achieve comparable accuracies. MobileNetV2 emerged as the optimal choice for contexts requiring smaller model size, while still achieving reasonable accuracies, albeit with training times similar to GoogleNet.

When tested on the CIFAR-10 dataset, we noticed a higher tendency for overfitting in ResNet18, underlining the need for careful consideration of the model's environment and the introduction of more robust regularization strategies.

In conclusion, this research underscores the importance of considering the specific requirements and constraints of a given task when selecting a model for deployment. Factors such as computational cost, model complexity, accuracy requirements, and the model's capacity to generalize are all integral to this decision-making process. While each model has its strengths, no single model emerged as universally optimal across all tasks and datasets. Therefore, the selection of a suitable model must be a nuanced process that carefully weighs these trade-offs to ensure effective and efficient performance on real-world tasks. It is our hope that the insights derived from this research provide valuable guidance in navigating these decisions.

REFERENCES

[1] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[2] Sandler, M. et al. (2019) MobileNetV2: Inverted residuals and linear bottlenecks, arXiv.org. Available at: https://arxiv.org/abs/1801.04381 (Accessed: 05 May 2023).

[3] He, K. et al. (2015) Deep residual learning for image recognition, arXiv.org. Available at: https://arxiv.org/abs/1512.03385 (Accessed: 9 May 2023).

[4] Szegedy, C. et al. (2014) Going deeper with convolutions, arXiv.org. Available at: https://arxiv.org/abs/1409.4842 (Accessed: 9 May 2023).
[5] Krizhevsky, A.; Sutskever, I. Hinton, G. E. (2012), ImageNet Classification with Deep Convolutional Neural Networks, in F. Pereira; C. J. C. Burges; L. Bottou K. Q. Weinberger, ed., 'Advances in Neural Information Processing Systems 25' , Curran Associates, Inc., , pp. 1097–1105 .
[6] Ioffe, S. and Szegedy, C. (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv.org. Available at: https://arxiv.org/abs/1502.03167 (Accessed: 5 May 2023).
[7] LeCun, Y. Cortes, C. (2010), 'MNIST handwritten digit database'.
[8] Krizhevsky, A. (2009) Learning multiple layers of features from tiny images, CIFAR-10 and CIFAR-100 datasets. Available at: https://www.cs.toronto.edu/ kriz/cifar.html (Accessed: 10 May 2023).
[9] Howard, A.G. et al. (2017) MobileNets: Efficient convolutional neural networks for Mobile Vision Applications, arXiv.org. Available at: https://arxiv.org/abs/1704.04861 (Accessed: 05 May 2023).
[10] Orhan, A.E. and Pitkow, X. (2018) Skip connections eliminate singularities, arXiv.org. Available at: https://arxiv.org/abs/1701.09175 (Accessed: 10 May 2023).
[11] Mallat, S. (2016) 'Understanding deep convolutional networks', Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), p. 20150203. doi:10.1098/rsta.2015.0203.
[12] Huang, G. et al. (2018) Densely connected Convolutional Networks, arXiv.org. Available at: https://arxiv.org/abs/1608.06993 (Accessed: 10 May 2023).
[13] Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press.
[14] Santos, C.F. and Papa, J.P. (2022) 'Avoiding overfitting: A survey on regularization methods for Convolutional Neural Networks', ACM Computing Surveys, 54(10s), pp. 1–25. doi:10.1145/3510413.
[15] Tensorboard: TensorFlow. Available at: https://www.tensorflow.org/tensorboard (Accessed: 6 May 2023).
[16] Antos, A., Devroye, L. and Gyorfi, L. (1999) 'Lower bounds for Bayes error estimation', IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(7), pp. 643–645. doi:10.1109/34.777375.