Image Recognition Through CNNs: A deep learning experimentation

Santiago Reyes Vargas – a1865001
The University of Adelaide 2023

SantRV/CIFR10_CNN: Convolutional Neural Network CFR10 (github.com)

Abstract

This report delves into the development process of a Convolutional Neural Network (CNN) algorithm for image recognition and classification. Neural network techniques are applied and adapted to the CIFAR10 dataset which consists of 60,000 32x32 colour images with 10 classes, leading to a proposed CNN model with an accuracy of over 80% across validation and testing data.

The conducted research starts with a literature review on the basic and more complex CNN architectures followed by a detailed analysis on the methodology undertaken by this author where three different CNN versions are proposed. While the same categorical cross-entropy loss function was used across the proposed models, these CNNs vary on their layer's depth, optimiser used and data normalisation and regularisation techniques.

The results of this experimentation concluded that the absence of regularisation techniques such as dropout led to an overfitted model with an accuracy of 99.77% in the training set and 81.55% in the validation set. Similarly, it was found that increasing the number of convolutional filters in the middle layer of a CNN architecture from 128 to 256 did not generated significant differences on the final loss and accuracy scores, accounting for a minimum validation loss of 0.4303 and 0.481 and an accuracy of 0.874 and 0.8583 respectively.

1. Introduction

With the bourgeoning amount of available unstructured data such as images, videos and audio files, the application of deep learning onto such datasets is leading to novel solutions such as self-driving cars, disease detection and speech recognition. A subset of Deep Neural Networks (DNNs) is Convolutional Neural Networks (CNNs) which consists of multiple convolution layers that extract low, mid, and high-level input features for better accuracy, making them a well-known architecture in the computer vision field.

CNNs are characterized by the use of convolution operations specified by the size and values contained within a filter. Filters are small matrices with a size of 2x2 up to 5x5, these serve as a sliding window placed over an input image to map to an output matrix. These filters scan a portion of the image and compute the weighted sum along with a bias term which allow the network to control the activation level of each feature map produced by the convolution, Figure 1. This process is followed by the use of an activation function to introduce non-linearity into the model and a pooling layer to reduce the special dimensions of the feature maps [1].

$$I_2(x,y) = \sum_{a,b} I_1(x+a, y+b)F(a,b) + B$$

$$I_1 = Input\ image$$

$$I_2 = Output\ image$$

$$F = Convolutional\ filter/kernel$$

$$B = Bias\ scalar\ term$$

Figure 1 – Convolutional operation equation (2D input).

Research has shown that increasing the number of layers in CNN models tends to lead to higher accuracy as is the case of ResNet-50 and MobileNet each with 50 layers and a top-1 accuracy of 75.3% for the former, and 28 layers and accuracy of 70.9% for the latter [2]. However, it is also clear that computational cost and training time tend to increase proportionally with the network's depth and that the use of regularisation and normalisation techniques, and different activation functions can have a direct impact on a model's performance.

Therefore, this report explores and analyses three proposed CNN architectures for image recognition applied to the CIFAR-10 dataset comprised of 32x32 images and 10 classes, Figure 2. Thereby, changes at the layer and parameter level are compared and examined to determine the existing trade-offs at each model iteration.

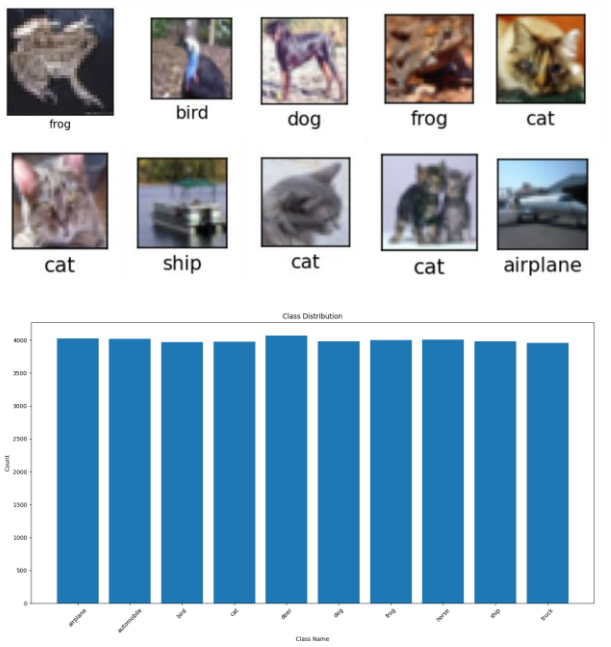

Figure 2 – CIFAR-10 dataset sample of images and classes.

In addition, this paper provides a critical review of CNN solutions, evaluating their loss and accuracy scores over 120-epochs each with a batch of 64. The presented results can inform researchers and students on the architecture design of deep learning models.

2. Literature Background

Convolutional Neural Networks (CNNs) have emerged as a transformative force in image recognition and classification, reshaping the landscape of computer vision. Originating in the early 1990s with Yann LeCun's pioneering work [3], CNNs have become the cornerstone of modern image analysis. These neural networks excel in automating the extraction of intricate image features and modelling spatial relationships, making them indispensable for tasks such as image categorization, object detection, and image segmentation.

A significant stride in the realm of image recognition is the concept of transfer learning. CNNs pre-trained on vast datasets have proved highly effective when fine-tuned for specific image classification tasks. Notable architectures such as VGG [4], GoogLeNet [5], and ResNet [6] have demonstrated the power of transfer learning, allowing researchers and practitioners to harness existing pre-trained models for rapid development of image recognition systems.

The horizon of CNNs for image recognition and classification is continuously expanding. These networks are finding applications in various domains, from medical image analysis to autonomous vehicles [7] and beyond. Reinforcement learning, in conjunction with CNNs, is making inroads in training agents for complex tasks [8]. With more intricate and powerful CNN architectures on the horizon, the future promises to deliver image recognition and classification systems that set new standards for accuracy and performance in computer vision.

3. Methodology

To explore the impact on accuracy, training time, and computational complexity three CNN models with different network depth, optimiser functions and regularisation and normalization layers were developed using TensorFlow and Keras. In this section, the architecture of the models is described, followed by an in-depth analysis on section 4 of their performance.

3.1. Data Pre-processing

The total size of the dataset is 60,000 32x32x3 images which were split into a training, validation and testing set in the following split: 70, 15, 15 percent respectively. The data labels were one-hot encoded into the 10 available classes.

3.2 Convolutional Neural Network Models

Three CNN models were developed based on the network architecture proposed in the Deep Learning Fundamentals lectures and were modified based on the conclusions drawn from the literature review. Overall, 2D convolutional layers, max pooling and fully connected layers are used as the core of the networks with 3x3 filters that allow the combination of information across space and channels. Also, zero padding (same padding) and a stride of 1 is used leading to an output image with the same size as the input, and the Rectified Linear Unit (ReLU).

Name	Optimiser	Regularization	Normalisation	Architecture
Basic CNN	Adam	None	None	Max 128 neurons on a single layer
CNN128	Stochastic Gradient Descend	Dropout	Batch Norm	Max 128 neurons on a single layer
CNN256	Adam	Dropout	Batch Norm	Max 256 neurons on a single layer

3.1.1. Basic CNN – Version 1:

The basic CNN model is the most basic model with one input layer, five hidden layers, and one fully connected output layer, Figure 4. This model does not implement regularisation, nor normalisation techniques.

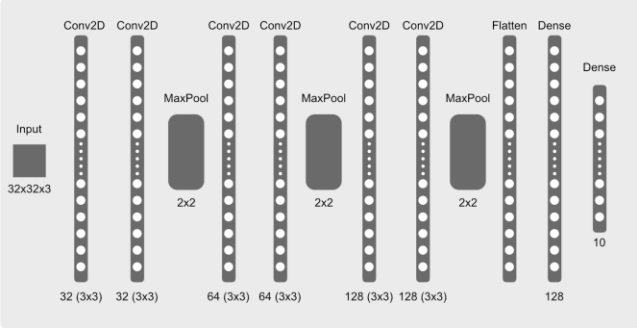

Figure 3 – Basic CNN (Version 1) model architecture.

3.1.2. CNN128 – Version 2:

This CNN network incorporates batch normalisation and dropout for regularisation with a total of 9 layers, including one 32x32x3 input layer, two Conv2D 32 layers, two Conv2D 64, two Conv2D 128, one flatten, one Dense 128 and one output Dense layer with 10 neurons corresponding to each class. While the batch normalisation steps are incorporated after each Conv2D, the Dropout is appended after each Maxpooling operation and one prior the output layer, Figure 4.

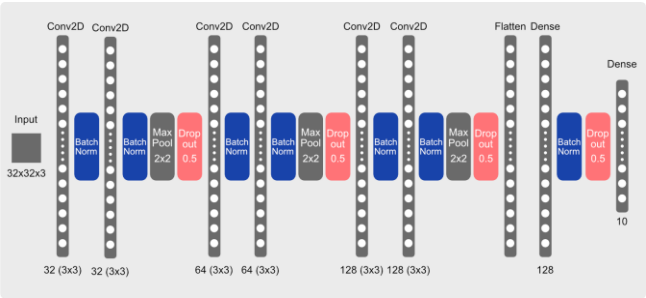

Figure 4 – CNN128 (Version 2) model architecture.

This model implements Stochastic Gradient Descend (SGD) as the optimiser to explore the impact of different optimisers in similar CNN networks.

3.1.3 CNN256 – Version 3

This CNN model builds on CNN128, thus adopting its layers, regularisation and normalisation techniques and extending them by adding two additionally layers each with 256 neurons, Figure 5. This is the deepest network tested on this research project and was aimed at determining the impact of increasing the number of neurons in terms of accuracy and training time.

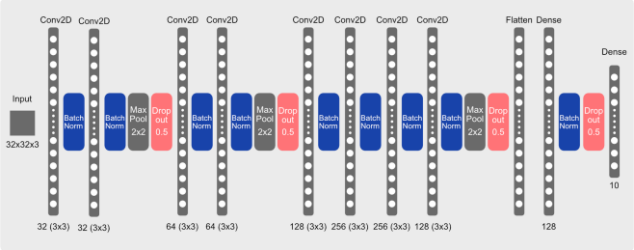

Figure 5 – CNN256 (Version 3) model architecture.

CNN256 implements the Adam optimiser which is characterised by having a learning rate that adapts during training for each parameter individually based on past gradient information, thus leading to faster convergence.

3.2 Model Evaluation

3.2.1 Category Cross Entropy

Given that this model aims to predict a discrete value that is within the range of 10 available classes, the category cross entropy loss was set as the criterion or loss function for the evaluation of these models. It measures the similarity between the predicted probabilities and the true binary labels.

$$L(y,p) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij} * \log(p_{ij})$$

$$N = Num\ data\ points$$

$$K = Num\ classes$$

$$y_{ij} = 1\ if\ class\ label\ for\ sample\ i\ is\ class\ j$$

$$p_{ij} = predicted\ prob\ to\ class\ j\ for\ sample\ i$$

Figure 6 – Categorical Cross-entropy loss function.

3.2.2 Stochastic Gradient Descent

This optimization function updates the model's parameters which comprises the weights and biases based on a random subset of the training data instead of the entire dataset. It seeks to find the set of parameters that minimizes the binary cross entropy function.

$$\theta(t+1) = \theta(t) - \varepsilon * \nabla J(\theta; x(i), y(i)))$$

$$\theta(t) \rightarrow Model's\ parameteres\ at\ time\ 't'$$

$$\varepsilon \rightarrow Learning\ rate$$

$$\nabla J\big(\theta; x(i), y(i)\big)$$
$$\rightarrow Gradient\ of\ loss\ J\ to\ gradients\ \theta$$

Figure 7 – Stochastic Gradient Descend equation.

3.2.2 Adam Optimiser

The Adaptive Moment Estimation (Adam) optimiser combines the first and second moment estimates to adaptively adjust the learning rate for each parameter. While the first moment estimate (Mt) can be regarded as a moving average of gradients, the second moment estimate (Vt) acts as a moving average of squared gradients, Figure 8.

$$g_t = Graddient\ at\ step\ t$$

$$\theta_t = Model\ weights\ and\ biasses\ at\ step\ t$$

$$\alpha = Learning\ rate$$

$$\beta_1 = Exponential\ decay\ rate\ for\ first\ moment\ estimate$$

$$\beta_2 = Exp\ decay\ rate\ for\ second\ moment\ estimate$$

$$First\ Moment(m_t) = \beta_1 * m_{t-1} + (1-\beta_1) * g_t$$

$$Second\ Moment(v_t) = \beta_2 * v_{t-1} + (1-\beta_2) * (g_t^2)$$

Bias correction to account for initialisation:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Update model parameters:

$$\theta_{t+1} = \theta_t - \alpha * \frac{\widehat{m_1}}{\sqrt{\hat{v}_t + \varepsilon}}$$

Figure 8 – Adaptive Moment Estimation equation.

The used Adam optimiser had a learning rate of 0.001, beta_1 = 0.9, beta_2 = 0.999, and epsilon = 1e-07.

4. Experiments & Analysis

All models were trained with 120 epochs and a 64-batch size. Early stopping was not implemented in order to analyse the asymptotical behaviour of the loss curve further discussed in this section. The results are summarised on Figure 9, and the accuracy on the validation set is depicted on Figure 10 and 11.

Model	Val Loss	Val Accuracy	Training Time(hrs)	Test Accuracy
1. Basic CNN	0.707	0.815	8.74	0.808
2. CNN128	0.430	0.874	5.62	0.871
3. CNN256	0.471	0.858	13.31	0.855

Figure 9 – Results summary of the three proposed CNNS

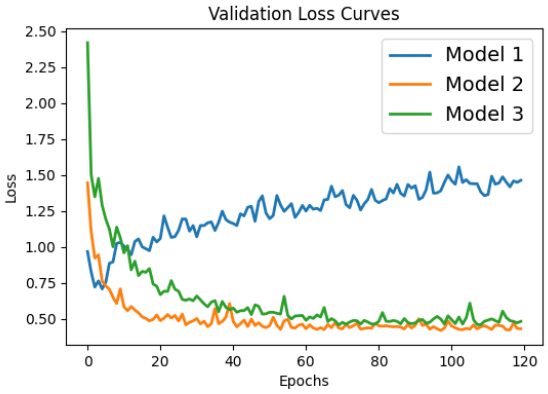

Figure 10 – Loss over epochs for all CNN models.

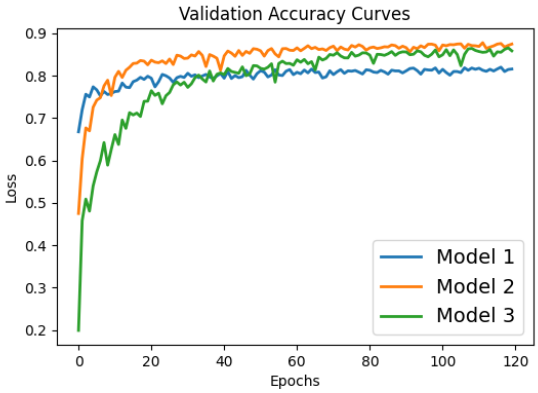

Figure 11 – Accuracy over epochs for all CNN models.

4.1. Model performance analysis

Overall, the three proposed models achieved an accuracy of over 80% in both the validation and testing sets, Figure 11. However, as seen on Figure 12, the Basic CNN model experienced overfitting and its loss started to increase from epoch 8 onwards; this undesirable situation suggests that the model performance is deteriorating and moving away from the optimal solution.

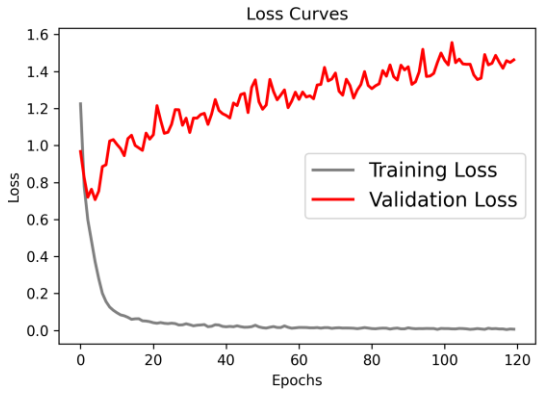

Figure 12 – Basic CNN loss curve, 120 epochs, 64 batch size.

Given that the Basic CNN model did not have any regularisation layers, it can be argued that the model started to fit the training data too closely, capturing the noise within and not generalising well. In fact, the Basic CNN model had an accuracy of 99.77% in the training and a final loss of over 1.4 indicating overfitting.

On the other hand, the CNN128 and CNN256 did not overfit and were generalised with the use of batch normalisation and dropout. While these models achieved a similar accuracy of 87.1% and 85.5% respectively, their main difference lies on the number of neurons in the central layer of the network.

Based on the obtained loss and accuracy results, it can be argued that by increasing the number of neurons from 128 to 256 in the central layer the accuracy did not have a significant improvement, yet the training time increased by 65.66%. Likewise, it is the case that CNN256 converged around epoch 80 while CNN128 converged at epoch 60, hence, the CNN256 architecture does not provide a valuable trade-off between the final model accuracy and its training time.

4.2. The impact of regularisation and normalisation layers

To address the overfitting issue identified in the Basic CNN model, on the second experiment conducted that led to the development of CNN128 the use of a Dropout technique for regularisation was implemented given that it has become a standard across numerous CNN implementations. This technique randomly deactivates a fraction of neurons during each forward and backward pass across the network, thereby adding noise to the network helping the model to learn more generalised and robust representations of the data [9].

Moreover, the use of batch normalisation techniques was incorporated on the subsequent CNN implementations which helps to reduce the risk of vanishing gradients leading to faster and more stable training [9]. This normalisation applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1, thus ensuring that the activations have a consistent distribution throughout the training process, Figure9.

$$Minibatch\ mean(\mu_B) = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$Minibatch\ variance(\sigma_B^2) = \frac{1}{m,}\sum_{i=1}^{m} (x_i - \mu_B)^2$$

$$Norm\ minibatch(\hat{x}_i) = \frac{x_1 - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$Output\ of\ ith\ neuron(y_i) = \gamma\hat{x}_i + \beta$$

$$\epsilon = Small\ constant\ to\ avoid\ division\ by\ zero$$

$$\gamma\ and\ \beta = Scaling\ and\ shifting\ params\ learned\ training$$

Therefore, unlike the overfitted Basic CNN model, the CNN128 and CNN256 had dropout and batch normalisation layers across the network. While a dropout layer with a probability of 0.5 was added at the end of each max-pooling layer, and before the final fully connected layer; the batch normalisation layers were incorporated after each convolutional (Conv2D) or Dense layers.

The impact of incorporating normalisation and regularisation layers can not only be seen in the training time of the models, but also in the asymptotic behaviour of the loss function curve. While Basic CNN took approximately 8.74 hours to train, CNN128 took 5.62 hours. That is an improvement of 35.69% considering that both networks had the same core layers and number of neurons in each.

Likewise, as seen on Figure 13, the asymptotic behaviour CNN128's loss function curve presents to be more stable and to converge at around epoch 60 with an approximate score of 0.435. Likewise, the accuracy of CNN128 seems to converge at epoch 80 reaching 87%.

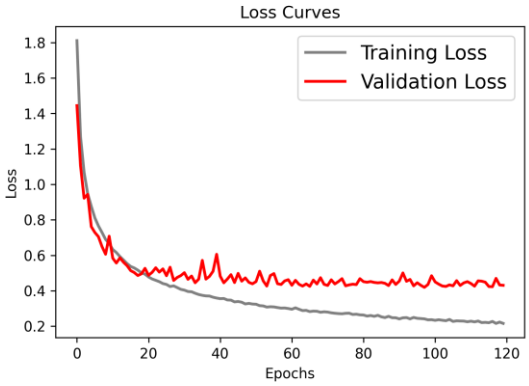

Figure 13 – Loss over epochs for all CNN128 model.

4.2. The impact of optimisation function.

While the Basic CNN and the CNN128 used the Adam optimiser, the CNN256 used Stochastic Gradient Descend (SGD) optimiser. From the literature review it can be inferred that the convergence time and generalisation of the model can be affected by the selected optimisation function.

By comparing results of the CNN128 and CNN256 models presented in Figure 5, it can be argued that a contributing factor in the training time was the use of SGD in the CNN256 which by using a fixed learning rate of 0.01 could had slowed the convergence time. On the other hand, CNN128 by utilising the Adam optimiser could benefit from a learning rate scheduling mechanism and a learning rate adjusted for each parameter individually based on the past gradient information, thus allowing it to adapt to the geometry of the loss function surface.

5. Conclusion

In this study, three CNN models were assessed for image recognition and classification. The Basic CNN model, lacking regularization, suffered from overfitting, hindering its performance. In contrast, CNN128 and CNN256, aided by batch normalization and dropout, mitigated overfitting, achieving validation accuracies of 87.1% and 85.5%, respectively. While CNN256 featured more neurons, it didn't offer a significant accuracy improvement despite a 65.66% longer training time.

The impact of regularization and normalization layers was substantial, as CNN128 exhibited shorter training times and stable loss convergence around epoch 60. Incorporating batch normalization and dropout proved effective.

Optimization function choice played a role, with CNN256's SGD leading to slower convergence. CNN128's dynamic learning rate adjustment with the Adam optimizer allowed it to adapt better to the loss function surface.

In summary, effective regularization, normalization, and optimization is vital for CNN performance. Striking a balance between complexity, training time, and performance is critical. Further refinement of these techniques can lead to efficient and accurate CNN models for image recognition.

5.1. Ideas for future work

Building on the experimental evidence presented in this report, several promising directions for future research and improvement in the field of image recognition using Convolutional Neural Networks (CNNs) can be considered:

Hybrid Architectures: Investigate the potential of hybrid CNN architectures that combine the strengths of different models. For instance, a model could leverage the regularization techniques of CNN128 with the number of neurons found in CNN256, aiming to strike a balance between training time and performance. Exploring other novel architectures and their impact on model convergence and accuracy is also an avenue for research.

Advanced Regularization: Experiment with advanced regularization techniques beyond dropout, such as L1 and L2 regularization, to further enhance model generalization and reduce overfitting. Comparing the efficacy of different regularization methods and their interactions with other layers can provide valuable insights.

Optimization Function Selection: Continue the exploration of optimization functions and learning rate schedules. Investigate the benefits of combining optimization functions, dynamically selecting them during training, or adapting the learning rate for each parameter individually. Assess their impact on convergence, model stability, and final accuracy.

Transfer Learning: Expand the analysis to include transfer learning with pre-trained CNN models. Investigate the advantages and trade-offs of fine-tuning pre-trained models for specific image recognition tasks, mainly when dealing with limited labelled data.

References

[1] I. Zafar, G. Tzanidou, R. Burton, N. Patel and L. Araujo, Hands-on Convolutional Neural Networks with TensorFlow: Solve computer vision problems with modelling in TensorFlow and Python, Mumbai: Packt Publishing Ltd. , 2018.

[2] A. Munir, J. Kong and M. A. Qureshi, Accelerators for Convolutional Neural Networks, New Jersey: John Wiley & Sons, Inc. https://doi.org/10.1002/9781394171910, 2023.

[3] Y. LeCun, L. Bottou , Y. Bengio and P. Haffner , "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 11, no. 86, pp. 2278-2324, 1998.

[4] K. Simonyan and A. Siserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv,* no. 1409.1556, 2014.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov and A. Rabinovich, "Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition," pp. 1-9, 2015.

[6] K. He, X. Zhang, S. Ren and J. Sun , "Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)," pp. 770-778, 2016.

[7] L. Wang, K. Wang, Y. Liu and Y. Lv, "Autonomous vehicle trajectory prediction with deep learning: A survey. IEEE Transactions on Intelligent Transportation Systems," pp. 1-21, 2021.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemere and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature,* vol. 518, no. 7540, pp. 529-533., 2015.

[9] R. O. Ogundokun, R. Maskeliunas, S. Misra and R. Damasevicius, "Improved CNN Based on Batch Normalization and Adam Optimizer," *International Conference on Computational Science and Its Applications,* no. https://link.springer.com/chapter/10.1007/978-3-031-10548-7_43, 2022.

[10] A. M. Egam and S. F. Dinneen, "Diabetes: Basic Facts - What is diabetes?," *Medicine,* vol. 50, no. 10. https://www.sciencedirect.com/science/article/pii/S1357303922001797, pp. 615-618, 2022.

[11] J. A. Luchsinger, "Type 2 Diabetes and Related Conditions in relation to dementia: An opportunity for prevention?," *Journal of Alzheimer's Disease,* vol. 20, no. 11. https://content-iospress-com.eu1.proxy.openathens.net/download/journal-of-alzheimers-disease/jad091687?id=journal-of-alzheimers-disease%2Fjad091687, pp. 723-736, 2010.

[12] WHO, "Diabetes," 5 April 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/diabetes. [Accessed 10 September 2023].

[13] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu and L. Shao, "Normalization techniques in training DNNs: Methodology, analysis and application," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE,* vol. 45, no. 8. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10056354, p. 10173, 2023.

[14] H. Malmgren , M. Borga and L. Niklasson, Artificial neural networks in medicine and biology, New York: Springer. https://link.springer.com/book/10.1007/978-1-4471-0513-8, 2000.

[15] M. Alssema, D. Vistisen, W. M. Heymans, G. Nijpels, P. Z. Zimmet, P. Z. Shaw and M. Elisson, "Risk scores for predicting type 2 diabetes: using the optimal tool," *PubMed,* no. https://pubmed.ncbi.nlm.nih.gov/21660635/, 2011.

[16] K. Abnoosian, R. Farnoosh and M. Behzadi, "Prediction of diabetes disease using an ensemble of machine learning multi-classifier models," *BMC Bioinformatics,* no. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-023-05465-z#:~:text=Prediction%20of%20diabetes%20disease%20using%20an%20ensemble%20of,...%204%20Machine%20learning%20models%20Single%20models%20, pp. 10-94, 2021.

[17] J. Yang and F. Wang, "Auto-ensemble: An adaptive learning rate scheduling based deep learning model ensembling," *IEEE,* no. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9274468, 2020.