Comparative Analysis of EfficientNet, ResNet, and Custom Model Architectures for Image Classification Tasks

Kirishoban Sanjeevvijay, URN: 6839353, ks02156@surrey.ac.uk

Abstract

This study explores the performance of various deep learning architectures, including ResNet and EfficientNet variants, along with custom model designs, for the task of image classification on a dataset comprising 9928 knife images in 192 classes. Extensive experimentation involves the variation of critical hyperparameters such as batch size, learning rate, weight decay, and optimizers to evaluate their impact on model performance. Through meticulous analysis and evaluation of mean Average Precision (mAP) and training loss metrics, EfficientNet B3 emerges as a standout performer, surpassing other models. The investigation also extends to modifications in architecture design, highlighting the significant superiority of established models over simpler custom architectures. Findings suggest potential avenues for model refinement and future research, emphasizing the importance of a comprehensive exploration of hyper-parameters and model architectures for enhanced image classification tasks.

1. Introduction

1.1. Background

The realm of computer vision has witnessed a profound transformation owing to advancements in deep learning and neural network architectures. This transformation is propelled by the evolution of various convolutional neural network (CNN) models, such as ResNet (Residual Network) and EfficientNet, each designed with distinctive features and architectural innovations. CNNs are at the core of image recognition tasks, extracting hierarchical features from images and enabling machines to comprehend, classify, and detect objects within them. Understanding the foundational principles and intricacies of these architectures is pivotal in comprehending their effectiveness and applicability in diverse tasks related to computer vision.[3] This study primarily focuses on a training dataset comprising 9,928 images categorised into 192 classes, alongside a test dataset containing 351 images.

1.2. Research Objectives

The primary objective of this study is to conduct an extensive comparative analysis of various convolutional neural network architectures, including EfficientNet (variants B0 to B3), ResNet (variants 18, 34, and 50), and a custom-designed model. The research aims to evaluate the performance of these architectures concerning image classification tasks, specifically focusing on their training loss, and mAP on test data.

This study will additionally investigate the impact of varying hyper-parameters, including learning rate, number of training epochs, batch size and weight decay, to discern the optimal configuration for the neural network models under examination. While investigating the outcome of changing a particular hyper-parameter, the values of all other hyper-parameters will be kept constant to avoid any ambiguity in the cause of the results. EfficientNet B0 will be the model used for the exploration of various hyper-parameter values.

2. Exploration of hyper-parameters

2.1. Batch size

Batch size refers to the number of training samples utilised in a single iteration to update the model's weights. A larger batch size processes more samples per iteration which may influence the model's ability to generalise in exchange for computational efficiency.[5]

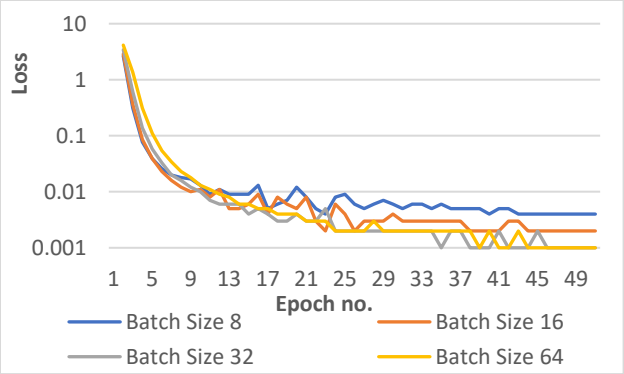

Figure 1: Illustrates the impact of altering the batch size on training loss across epochs, employing a logarithmic scale for enhanced visualization of values and trends. Lower training loss values are usually indicative of superior model performance.

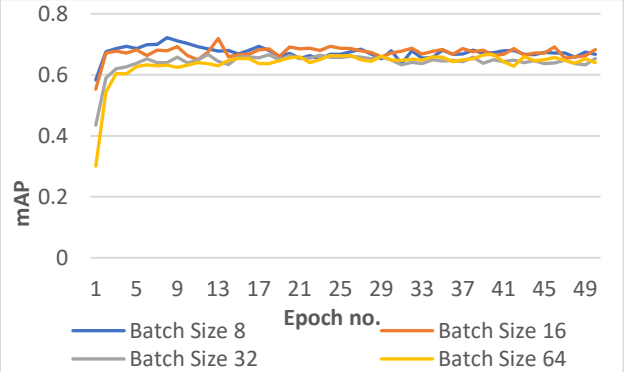

Figure 2: Effect of changing batch size on mAP with each epoch. A higher mean average precision (mAP) indicates a better performing model/parameter value on the test data and is therefore a great performance indicator.

In Figure 2, a noticeable pattern emerges as the batch size decreases: the mean average precision (mAP) tends to improve consistently across various epochs. This observation supports the theory of a smaller batch size improving generalisation. Surprisingly, Figure 1 presents a more intricate scenario; it does not depict a discernible trend of training loss decreasing in correspondence with the batch size. This observation suggests that while smaller batch sizes contribute to enhanced mAP, their impact on training loss may be more nuanced.

For the subsequent experiments, a batch size of 16 will be employed as a balanced value, aiming to strike a compromise between computational resource utilisation and the preservation of generalisation performance.

2.2. Learning rate

The learning rate determines the magnitude at which the model's weights are updated during training. A higher learning rate may facilitate faster convergence but risks overshooting the optimal weights, leading to instability or divergence. Conversely, a lower learning rate offers more cautious weight adjustments, potentially ensuring convergence but at the cost of prolonged training time and the possible risk of getting stuck in a local minimum.[6]

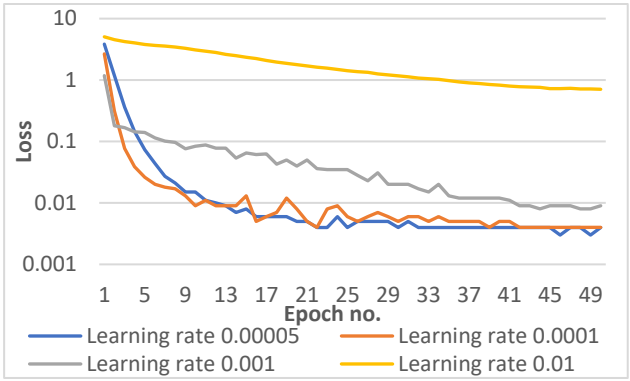

Figure 3: Effect of changing learning rate on training loss with each epoch. A logarithmic scale is used for loss throughout all graphs in this report.

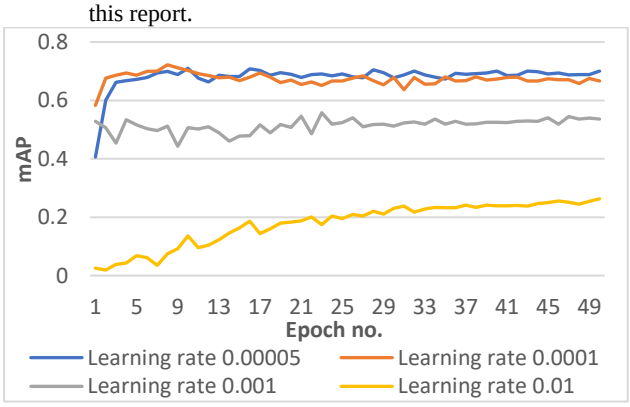

Figure 4: Effect of changing learning rate on mAP with each epoch.

From Figure 3, the increase in the learning rate corresponds to higher training loss, as predicted. However, with learning rates of 0.00005 and 0.0001, one causes more/less loss than the other, varying across epochs. This suggests that a learning rate lower than 0.0001 may offer minimal convergence benefits while demanding excessive computational time. Figure 4 supports these observations, showcasing an improvement in mAP with decreasing learning rates, although the distinction is again somewhat subtle when comparing rates of 0.00005 and 0.0001.

As a learning rate of 0.0001 shows very similar results to that of 0.00005 while being less computationally

expensive, a learning rate value of 0.0001 will be used for the remaining experiments.

2.3. Weight decay

Weight decay controls the extent of regularisation applied to the model's weights during training. Higher weight decay values might excessively penalize large weights, potentially improving generalisation but risking underfitting. On the other hand, lower weight decay values may permit larger weights, possibly leading to overfitting. Striking a balance between these extremes is crucial to optimise a model's performance and prevent overfitting or underfitting issues.[7]

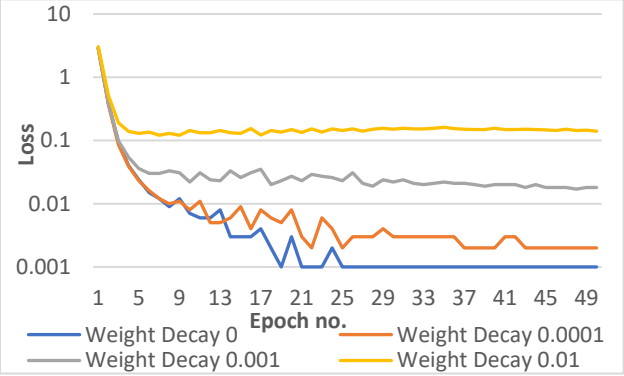

Figure 5: Effect of changing weight decay on training loss with each epoch.

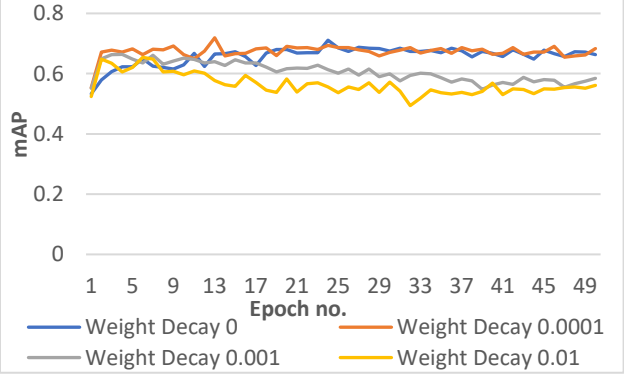

Figure 6: Effect of changing weight decay on mAP with each epoch.

In Figure 5, the trends become more distinct as training progresses: lower weight decay values correspond to lower observed losses, with a weight decay of 0 exhibiting the most favourable performance. This trend aligns with Figure 6, illustrating a consistent improvement in mAP as weight decay decreases across various epochs. Notably, the model employing a weight decay of 0.0001 consistently demonstrates mAP values similar to those with a weight decay of 0, irrespective of epoch number.

The results depicted in Figures 5 and 6 pertain specifically to the Adam optimizer. It's noteworthy that the impact of varying weight decay values differs significantly when considering alternative optimizers like SGD and AdamW, which are expanded upon in the subsequent section. For both SGD and AdamW optimizers, the most optimal performance is achieved with a weight decay value of 0.01, among those examined in this report.[8] Taking this into consideration, to compare the efficacy of the optimizers in Section 2.4, the best performing weight decay was chosen for each optimizer.

2.4. Optimizers

Optimizers are responsible for updating the model's parameters during the learning process. Adam, AdamW, and SGD are three popular optimization algorithms to be explored in this study.[9]

Adam combines adaptive learning rates for individual model parameters, offering fast convergence by adjusting the learning rates throughout training.[10]

AdamW, an extension of Adam, incorporates weight decay directly into its optimization, often leading to more stable convergence and robust performance.[11]

Stochastic Gradient Descent (SGD) iteratively updates model parameters by computing gradients for a subset of the training dataset.[12] A comparison of the optimizers using their respective best performing weight decay values (WD=0.01 for SGD, AdamW and WD=0 for Adam) can be seen in Figures 7 and 8.

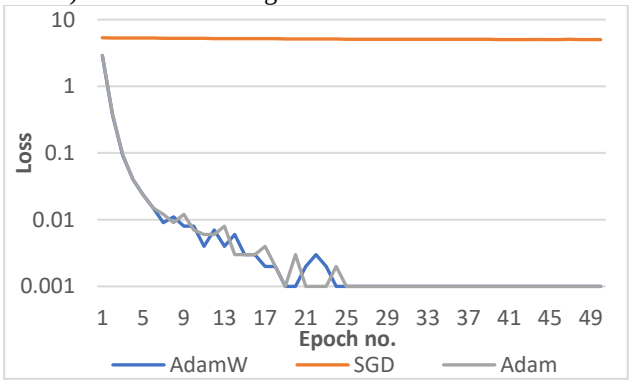

Figure 7: Effect of changing optimizers on training loss with each epoch.

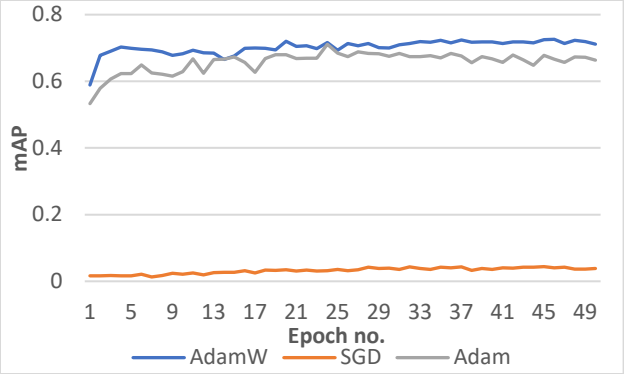

Figure 8: Effect of changing optimizers on mAP with each epoch.

In Figures 7 and 8, SGD demonstrates notably poorer performance, as evidenced by higher training loss and lower mAP compared to both Adam and AdamW. When comparing Adam and AdamW, Figure 7 reveals minimal differences in training loss trends across various epochs. However, AdamW consistently produces higher mAP values across all epochs up to epoch 50, exhibiting superior performance against the test data when compared to Adam.

Therefore, the most optimal hyper-parameter values identified thus far for EfficientNet B0 are as follows: batch size = 16, learning rate = 0.0001, optimizer = AdamW, weight decay = 0. These established values will be utilised henceforth to assess various deep model architectures and their respective variations.

3. Deep model architectures

3.1. ResNet architecture

Residual Network (ResNet) variations, such as ResNet-18, ResNet-34, and ResNet-50, which are explored in this study, differ primarily in their depth. ResNet-18 comprises 18 layers, while ResNet-34 has 34 layers; ResNet-34 being deeper than ResNet-18, allows it to capture more intricate features.

In contrast, ResNet-50 introduces bottleneck blocks that employ three convolutional layers: 1x1, 3x3, and 1x1.[2]

3.2. EfficientNet architecture

The key idea behind EfficientNet involves scaling the model's depth, width, and resolution using compound scaling: this is what primarily differs between EfficientNet variants from B0 to B7.[4] This study investigates EfficientNet variants from B0 to B3.

3.3. Testing

The models explored in this report were EfficientNet: B0, B1, B2, B3 and ResNet: 18, 34, 50. The image classification capabilities of these models on the knife classification datasets can be seen in Figures 9 and 10 below.

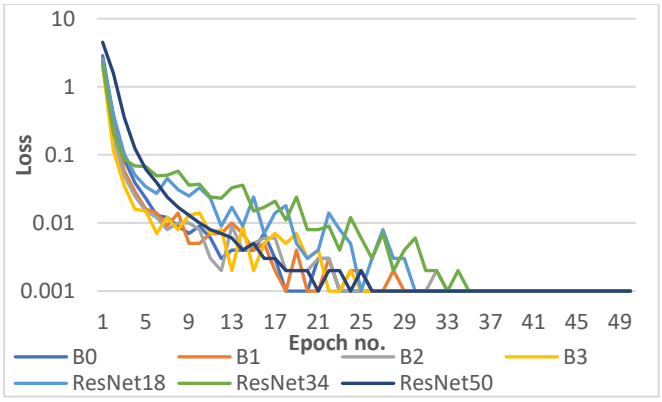

Figure 9: Training loss against each epoch for EfficientNet and ResNet model variations.

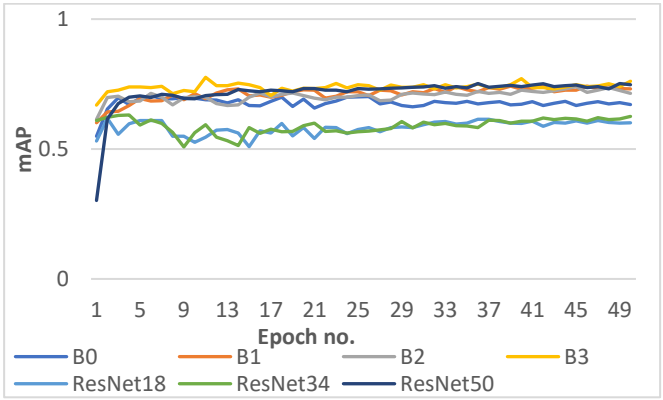

Figure 10: mAP against each epoch for EfficientNet and ResNet model variations.

Figures 9 and 10 reveal notable distinctions in performance among the different architectures. Both ResNet18 and ResNet34 demonstrate relatively inferior results in terms of mAP and training loss compared to the various EfficientNet variants. Meanwhile, ResNet50 exhibits superior performance in both metrics when compared to the other ResNet variations. Among the EfficientNet models, consistent training loss values are observed across epochs, and a similar trend is noticed in mAP, with EfficientNet B3 showcasing better mAP values compared to its counterparts. Additionally, Figure

10 demonstrates that EfficientNet B3's performance closely aligns with that of ResNet50.

The subtle discrepancies in performance among EfficientNet B0, B1, B2, and B3 could be attributed to maintaining the same image resolution across these models. This decision was necessitated by limited computational resources, and it's likely that altering image resolutions for each EfficientNet variation would yield much more varied and improved results.

3.4. Model customisation

Utilising EfficientNet B3 as the top-performing model architecture with the previously established hyper-parameters, subsequent experiments involved customising the existing B3 model architecture. The initial experiment focused on removing the final fully connected classification layer from the pre-trained model. Another experiment introduced an additional convolutional layer after the penultimate layer, aiming to augment the model's capacity for feature representation before the final classification layer. Subsequently, a final experiment was conducted using a custom model—a simplified CNN architecture. This model comprises three convolutional layers, each followed by max-pooling to downsample feature maps. The architecture culminates with two fully connected layers responsible for final classification.[13]

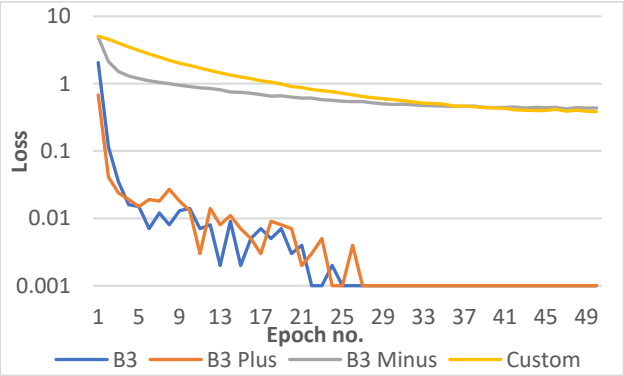

Figure 11: Training loss against each epoch for different model architecture variations. B3 denotes the standard EfficientNet B3 model architecture. B3 Plus denotes the model with the additional layer and B3 Minus represents the model with the final layer removed. Custom is the simple CNN architecture that was tested.

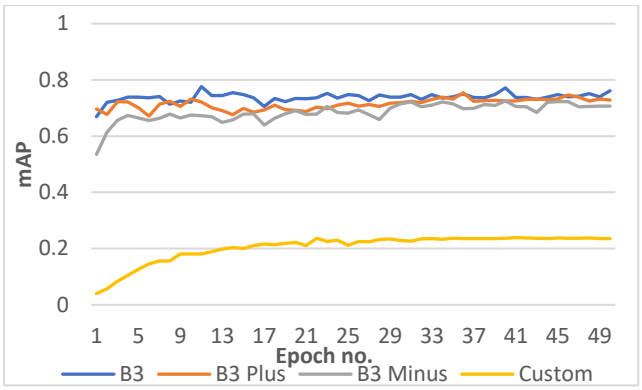

Figure 12: mAP against each epoch for different model architecture variations.

As anticipated, EfficientNet B3 notably outperforms the custom CNN model in both training loss and mAP (as shown in Figures 11 and 12). The stark performance contrast illustrates the superiority of the other models assessed in this study and emphasises the limitations of the simplistic custom CNN. Nevertheless, the custom CNN architecture might find utility in specific niche image classification tasks constrained by smaller datasets or severely limited computational resources.
Analysing alterations to the B3 architecture, the removal of the final classification layer led to a considerable increase in training loss, as anticipated, yet did not yield a significant decrease in mAP, particularly for higher epoch counts. Conversely, the introduction of an additional layer did not exhibit noticeable improvements in training error. In fact, when examining mAP, it slightly underperformed. However, it's noteworthy that the mAP is nearly on par with the original model, particularly in later epochs toward epoch 50.

4. Conclusion

4.1. Summary of findings

The investigation of batch size highlighted its impact on model convergence and generalisation, demonstrating a trade-off between computational efficiency and model performance. Learning rate adjustments showcased their pivotal role in steering model convergence, with lower rates yielding steadier convergence. Additionally, weight decay emerged as a crucial factor in preventing overfitting, impacting the model's generalisation ability significantly. The number of epochs was a significant factor up until approx. epoch 20 after which there were

very little changes seen for all hyper-parameters and model architectures.

The comparison across optimizers indicated distinct behaviours in training loss and mAP, with AdamW exhibiting superior performance among the explored optimizers. EfficientNet B3 consistently demonstrated superior performance compared to ResNet variations and custom CNN architectures, highlighting its robustness in image classification tasks. Furthermore, alterations to model architectures showcased nuanced effects on training loss and mAP, affirming the intricate relationship between model complexity and performance.

4.2. Limitations & future work

This study's exploration of hyperparameters, while extensive, might benefit from a broader scope encompassing a wider range of values and combinations. This could uncover unforeseen optimal configurations and offer deeper insights into model behaviour.

The adjustments made to model architectures were relatively basic, involving minor changes like adding or removing a single layer. Future investigations could delve into more substantial alterations, such as introducing new blocks, altering layer sequences, or adapting architectures more extensively.

The study primarily relied on mean Average Precision (mAP) as the primary evaluation metric. While informative, relying solely on mAP might limit the comprehensive assessment of model performance. Integrating multiple evaluation metrics, including accuracy measures, could offer a more holistic view of model functionality.

Expanding the exploration to encompass larger models, such as EfficientNet B7, could provide insights into scalability and performance, offering a deeper understanding of higher-capacity models.

Considering the concept of model ensembling by combining the best-performing models might enhance predictive performance and model robustness, warranting exploration in future studies.

Additionally, experimenting with more diverse datasets could elucidate how these models generalize across various data distributions, contributing to a deeper understanding of their performance in different scenarios.

Addressing these limitations and pursuing these avenues for future research would not only enhance the current study but also contribute to the broader field of image classification models. The majority of these limitations were due to time constraints and/or computational limitations which can easily be overcome in future experimentation.

References

[1] House of Commons Library (Oct. 2023), Knife crime statistics England and Wales. Available at: https://commonslibrary.parliament.uk/research-briefings/sn04304/ (01/12/23)

[2] Analytics Vidhya (Feb. 2023), Deep Residual Learning for Image Recognition (ResNet Explained). Available at: https://www.analyticsvidhya.com/blog/2023/02/deep-residual-learning-for-image-recognition-resnet-explained/ (01/12/23)

[3] Run.ai (2023), Deep Learning for Computer Vision. Available at: https://www.run.ai/guides/deep-learning-for-computer-vision#:~:text=Computer%20vision%20algorithms%20analyze%20certain,commonly%20used%20for%20computer%20vision. (01/12/23)

[4] Mingxing Tan, Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*, arXiv:1905.11946 [cs.LG].

[5] Medium (Jan. 2022), How does Batch Size impact your model learning. Available at: https://medium.com/geekculture/how-does-batch-size-impact-your-model-learning-2dd34d9fb1fa (01/12/23)

[6] Medium (Nov. 2023), Machine Learning: Learning Rate. Available at: https://baotramduong.medium.com/machine-learning-learning-rate-3398f7751efc (01/12/23)

[7] Maksym Andriushchenko, Francesco D'Angelo, Aditya Varre, Nicolas Flammarion. *Why Do We Need Weight Decay in Modern Deep Learning?*, arXiv:2310.04415 [cs.LG]

[8] Ilya Loshchilov, Frank Hutter, *Decoupled Weight Decay Regularization*, arXiv:1711.05101 [cs.LG]

[9] PyTorch (2023), torch.optim. Available at: https://pytorch.org/docs/stable/optim.html#algorithms (01/12/23)

[10] Machine Learning Mastery (January 2021), Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Available at: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (01/12/23)

[11] Fast.ai (July 2018), AdamW and Super-convergence is now the fastest way to train neural nets. Available at: https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html (01/12/23)

[12] Analytics Vidhya (Sept. 2023), A Comprehensive Guide on Optimizers in Deep Learning. Available at: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/ (01/12/23)

[13] Analytics Vidhya (May 2023), Build an Image Classification Model using Convolutional Neural Networks in PyTorch. Available at: https://www.analyticsvidhya.com/blog/2019/10/building-image-classification-models-cnn-pytorch/ (01/12/23)