# Implementing CNN with ResNet-18, EfficientNetB0 and AlexNet

Tanvi Krishna Murthy Jois
a1942914
The University of Adelaide
Adelaide, South Australia
a1942914@adelaide.edu.au

## Abstract

*Convolutional Neural Networks (CNNs) have become one of the most important aspects of machine learning and deep learning. When it comes to image analysis, CNNs have revolutionized the power of machines in understanding and automating the learning of visual features from data. This project involves the implementation and the comparison of three distinct CNN models: ResNet-18, EfficientNetB0 and AlexNet*

*The models are implemented on the CIFAR-10 dataset containing 60,000 images. The dataset is divided into test (10,000), train (40,000) and validation (10,000) subsets to evaluate the models. Various pre-processing techniques like normalization and augmentation are used to improve the model's predictability.*

*The models are trained using stochastic gradient descent (SGD) and L2 Regularization to prevent overfitting. Different learning rates were employed to the ResNet-18 model to optimize it for better performance.*

*The results pointed out that AlexNet, although a model with simple architecture, proved to be better than ResNet-18 and EfficientNetB0 with an accuracy of 81.5%.*

## Introduction

Complex tasks like image classification, computer vision, object detection and segmentation can be achieved by using the Convolutional Neural Networks (CNNs) [1]. CNNs are one of the most effective visual data networks in modern technology. Neural networks are derived from the neural networks in the biological sense and have three main pillars: convolution, pooling and hierarchical feature learning.

The CIFAR-10 dataset containing 60,000 images is in the size format of 32*32*3 (RGB, 32 pixels both in length and width) is used in this project. The dataset holds 10 classes: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship and Truck. Three CNN models are used in this project: ResNet-18, EfficientNetB0 and AlexNet.

The ResNet-18 model developed in 2015 is comprised of 18 layers. It is used for moderately large dataset but can be adapted to larger datasets with tuning. The concept of residual learning is inculcated in ResNet-18. This helps in reducing the vanishing gradient problem.

EfficientNetB0 model developed in 2019 utilizes compound scaling, a method where the model dimensions are scaled uniformly to improve performance. The EfficientNetB0 model has 16 layers and is used for very large-scale datasets.

AlexNet introduced in 2012 [2], is a model that is used for small-scale datasets. Its model has 8 layers with 5 convolutional layers and 3 fully connected layers. For CIFAR-10, AlexNet is a pioneering model.

The objective of this project is to analyze the performance of the three architectures on the CIFAR-10 dataset. The models are trained and evaluated on the same conditions such as data augmentation, L2 Regularization and Stochastic Gradient Descent with momentum.

This report aims to understand the strengths and weaknesses of each model on a relatively small dataset (CIFAR-10) and provide insights into how the architectural complexity of the models affects the performance of the models.

## Methodologies

The breakdown of the methodologies implemented in the project is given in this section. Each CNN architecture is optimized using different combinations of regularization, data augmentation and other training strategies to improve the model performance and generalization.

Few key functions used in this project in all three models are in the below sections

### Conv2D

Conv2D which helps in applying convolutional operations to extract features like edges and textures from the given data. The data is extracted using 64 filters. The general form of Conv2D is given by

$$y = Conv(x) + b$$

***Equation 1: Conv2D general formula***

Where,

- y is the output
- x is the input image
- Conv(x) is the convolution applied to x
- b is the bias term

## Batch Normalization

Batch normalization is then used to stabilize the dataset and encourage faster training by normalizing the activations. This helps the model to converge faster, and the generalization is also improved. As a result, overfitting of the model is reduced. The batch normalization is done with the help of the equation

$$\hat{z} = \frac{z - \mu}{\sigma}$$

***Equation 2: Batch normalization general equation***

Where,

- $\hat{z}$ is the output
- z is the activation from the previous layer
- $\mu$ is the mean of the activations
- $\sigma$ is the standard deviation of the activations

The normalized activation is then scaled and shifted.

## Activation function

The activation function "ReLU" is used for AlexNet and ResNet-18. This helps in encouraging the network to learn more complex patterns and prevents gradient vanishing problem. The negative values in the output are set to 0 to achieve this.

$$f(z) = \max(0, z)$$

***Equation 3: Activation function general equation***

Where,

- $f(z)$ is the output
- $z$ is the input value

## L2 Regularization

L2 Regularization is used to deal with very large weights in the model which can cause overfitting. The general equation of the regularization function is

$$L_{total} = L_{original} + \gamma \sum_{i=1}^{n} w_i^2$$

***Equation 4: L2 Regularization general formula***

Where,

- $L_{total}$ is the total loss with regularization
- $L_{original}$ is the original loss
- $\gamma$ is the regularization parameter
- $w_i$ is the weight of the i-th parameter
- n is the total number of weights

## Stochastic Gradient Descent (SGD) with momentum

The stochastic gradient descent uses "memory" from the pervious gradient to the next gradient helping stability of the model and faster convergence. The general equation for SGD with momentum is given by

$$v^{(t+1)} = \gamma . v^t + \mu . \nabla L(w^t)$$
$$w^{(t+1)} = w^t - v^{(t+1)}$$

***Equation 5: Stochastic Gradient Descent with momentum general equation***

Where,

- $v^t$ is the velocity at time t
- $\gamma$ is the term representing momentum
- $\mu$ is the learning rate
- $\nabla L(w^t)$ is the gradient loss function with respect to weights

## Data Augmentation

To improve generalization and overfitting issue, the project involves the usage of data augmentation. The data augmentation was used using ImageDataGenerator function. The dataset is subjected to various transformations like random rotations, horizontal flips, width and weight shifts and random zooms [3]. This exposes the model to various versions of the same data and helps the model become more robust improving its generalization capabilities. The general equation for data augmentation is

$$\bar{x} = T_n(T_{n-1}(\dots T_1(x) \dots))$$

***Equation 6: General equation for data augmentation***

Where,

- $\bar{x}$ is the augmented data
- $T_1 \dots T_n$ is the series of augmentation functions (rotation, flip, scaling etc.)

## ResNet-18

The ResNet-18 model is demonstrated in this project. The ResNet-18 uses a residual learning framework. This is a function which helps us handle the complexities of deeper architecture by mitigating the vanishing gradient issues. The general equation for the residual function[4]

was used. The general equation for the residual function is as follows

$$y = F(x) + x$$

***Equation 7: Residual function general equation***

Where,

- y is the output after residual connection
- x is the input tensor
- F(x) is the residual learned through different gradients

In the early stages of our project, The ResNet-18 model was overfitting. The training accuracy was high, but the validation accuracy became stagnant. This indicated poor generalization of the unseen data. To mitigate this issue, we used SGD, L2 regularization with different weights, and different learning rates (0.01, 0.001, 0.0001).

### EfficientNetB0

EfficientNetB0 is a model used to maintain computational efficiency. Although it is a lighter model compared to ResNet-18, in our project, EfficientNetB0 has shown competitive performance.

The SoftMax activation function is used in the EfficientNetB0 unlike the other two models. The EfficientNetB0 involving data augmentation and L2 Regularization proved to be effective in reducing the overfitting issue. The general equation for SoftMax activation functions is given by

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}}$$

***Equation 8: SoftMax activation function equation***

Where,

- $z_i$ Logit for class i
- $\hat{y}_i$ Predicted probability for class i.
- K: Total number of classes.

### AlexNet

AlexNet performed better than all the models in our project. Designed for small datasets, AlexNet has the best accuracy among all the other models for the CIFAR-10 dataset. Five convolutional layers are used in AlexNet and 3 fully connected layers are used. Fully Connected Layers with Dropout is the unique aspect of the AlexNet model in our project. This is given by the equation

$$y = x.M$$

***Equation 9:General equation for fully connected layers with dropout***

Where,

- y is the output
- x is the input activation (values before dropout)
- M is the binary mask

## Experimental Analysis

This section summarizes the experimental analysis of the project

### ResNet-18 initial model experimental results.

Initially ResNet-18 was used in the project. The L2 Regularization was set to 0.005 and data augmentation function was used to mitigate overfitting.

The accuracy for this was around 40% which is very low and the model was significantly overfitting. Although L2 regularization and data augmentation was applied, the model was overfitting.

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Airplane | 0.41 | 0.63 | 0.49 | 1000 |
| Automobile | 0.94 | 0.55 | 0.69 | 1000 |
| Bird | 0.19 | 0.87 | 0.31 | 1000 |
| Cat | 0.34 | 0.42 | 0.38 | 1000 |
| Deer | 0.60 | 0.18 | 0.27 | 1000 |
| Dog | 0.64 | 0.02 | 0.03 | 1000 |
| Frog | 0.65 | 0.35 | 0.45 | 1000 |
| Horse | 0.92 | 0.33 | 0.48 | 1000 |
| Ship | 0.87 | 0.13 | 0.23 | 1000 |
| Truck | 0.86 | 0.58 | 0.69 | 1000 |
| accuracy | | | 0.40 | 10000 |
| macro avg | 0.64 | 0.40 | 0.40 | 10000 |
| weighted avg | 0.64 | 0.40 | 0.40 | 10000 |

***Figure 1: Classification report of the initial ResNet-18 implementation***

The classification report suggested that classes like Automobile, Horse, Ship have high precision but low recall suggesting that the model is missing many true positives. The Bird class has high recall but low precision indicating that there are many false positives.


***Figure 2: confusion matrix for the initial experiment***

The confusion matrix clearly indicates the key issues in specific classes which might be causing the overfitting issue. This suggests that the model architecture needs to be tweaked.

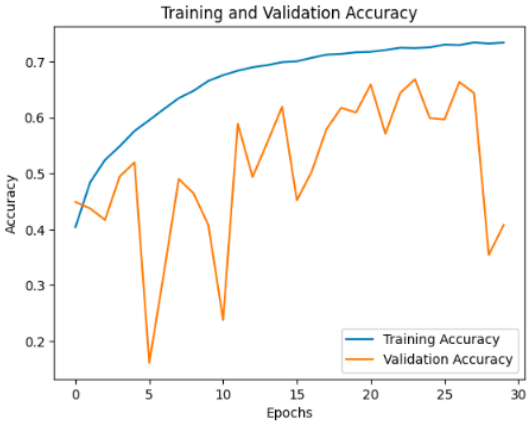


*Figure 3: Training and validation accuracy plot for initial model*

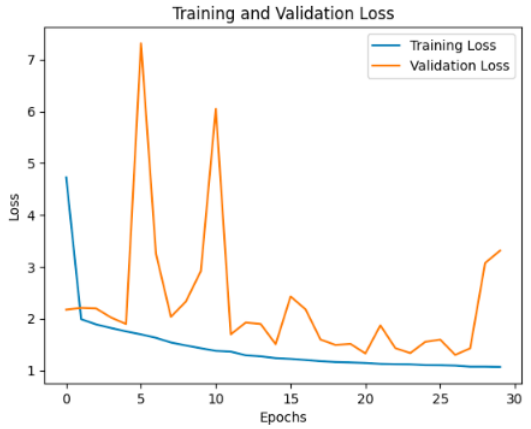


***Figure 4: Training and Validation loss graph for initial model***

The graphs for training vs validation accuracy and validation loss depicted a significant gap between the parameters indicating that the model was overfitting. The model initially used the "adam" optimizer and tested for 30 epochs.

The model was then subjected to a tweak with the optimizer and tested with different learning rates.

## ResNet-18 improved model experimental results.

The optimizer used to tweak the sequential model was stochastic gradient descent (SGD). When the stochastic gradient descent optimizer was used, the model successfully performed better without overfitting.

The L2 Regularization was lowered to 0.001 and the model was tested for 20 epochs with various learning rates namely 0.01, 0.001 and 0.0001. This improved the model performance significantly giving the accuracy of 80% for 0.01, 79% for 0.001 and 61% for 0.0001 learning rates.

This suggested that the 0.01 learning rate had the best balance between convergence and generalization while avoiding both overfitting and underfitting. The drop in the accuracy when 0.0001 learning rate was used indicates that the model is learning very slowly and is failing to converge on a solution in 20 epochs.

Classification Report for learning rate 0.01:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Airplane | 0.89 | 0.80 | 0.84 | 1000 |
| Automobile | 0.94 | 0.90 | 0.92 | 1000 |
| Bird | 0.94 | 0.53 | 0.68 | 1000 |
| Cat | 0.75 | 0.56 | 0.64 | 1000 |
| Deer | 0.76 | 0.76 | 0.76 | 1000 |
| Dog | 0.62 | 0.84 | 0.71 | 1000 |
| Frog | 0.84 | 0.85 | 0.84 | 1000 |
| Horse | 0.62 | 0.97 | 0.76 | 1000 |
| Ship | 0.93 | 0.88 | 0.91 | 1000 |
| Truck | 0.91 | 0.86 | 0.88 | 1000 |
| accuracy | | | 0.80 | 10000 |
| macro avg | 0.82 | 0.80 | 0.79 | 10000 |
| weighted avg | 0.82 | 0.80 | 0.79 | 10000 |

***Figure 5: Classification report for learning rate 0.01***

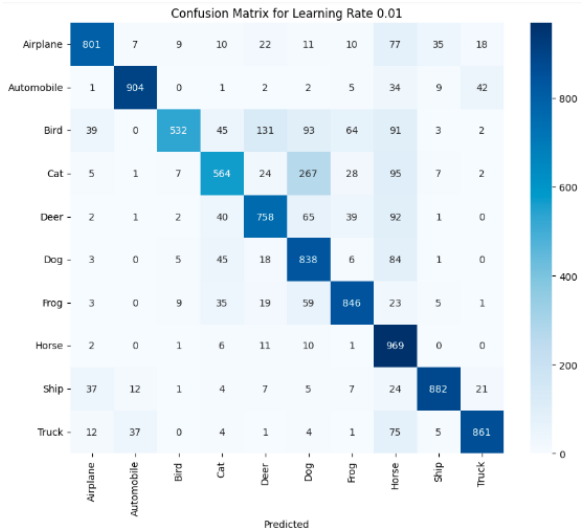


*Figure 6: Confusion matrix for learning rate 0.01*


*Figure 7: Training vs validation accuracy and validation loss plot for 0.01 learning rate in improved model*

## EfficientNetB0 experimental results

EfficientNetB0 model has a total of 16 layers. It uses compound scaling as its main function to produce efficient results. Compound scaling balances the model depth, width and resolutions to maximize efficiency. This can handle a large dataset and provide efficient results.

As the model computed the accuracy this project received for the CIFAR-10 dataset is 64.09%.

EfficientNetB0 failed to perform in this project because it is developed to handle extremely complex datasets[5]. This model, when applied to a significantly small dataset like CIFAR-10 The model underperforms and does not utilize its full potential. Furthermore, EfficientNetB0 is designed to handle high resolution data like 224*224 pixels. As the data in the CIFAR-10 dataset has only 32*32 pixels, it is not enough for EfficientNetB0 to draw a solution.

```
Classification Report:

              precision    recall  f1-score   support

    Airplane       0.72      0.66      0.69      1000
  Automobile       0.74      0.79      0.76      1000
        Bird       0.81      0.28      0.42      1000
         Cat       0.48      0.42      0.45      1000
        Deer       0.54      0.66      0.59      1000
         Dog       0.62      0.52      0.56      1000
        Frog       0.59      0.83      0.69      1000
       Horse       0.62      0.78      0.69      1000
        Ship       0.89      0.63      0.74      1000
       Truck       0.61      0.84      0.71      1000

    accuracy                           0.64     10000
   macro avg       0.66      0.64      0.63     10000
weighted avg       0.66      0.64      0.63     10000
```

*Figure 8: Classification report of EfficientNetB0 model on the CIFAR-10 dataset*

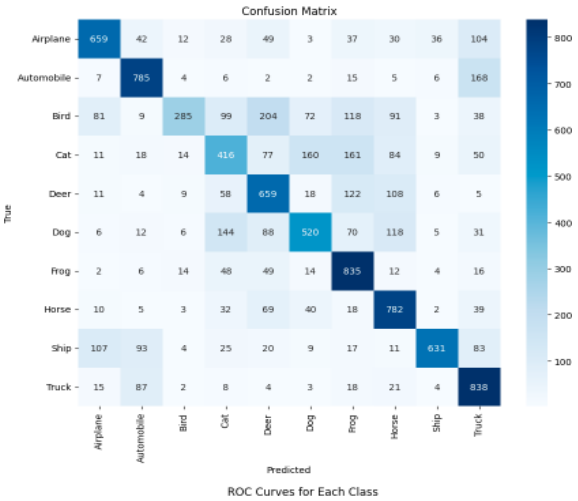


*Figure 9: Confusion matrix for EfficientNetB0*

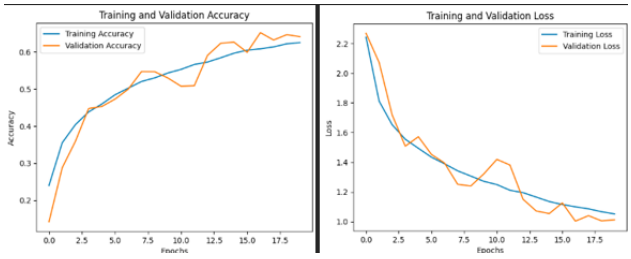


*Figure 10: Training vs validation accuracy and validation loss graphs*

## AlexNet experimental results

AlexNet, being a lightweight model takes in small datasets and comparatively works better for smaller datasets. It has 8 layers in total. 5 convolutional layers and 3 fully connected layers. The AlexNet model performed better than all other models for the CIFAR-10 dataset. The accuracy for the AlexNet model was 81.5%

The AlexNet model performed better because it was compatible with the CIFAR-10 dataset as the dataset is small and the AlexNet model performs well with smaller datasets as the model is less in complexity and the simpler architecture results in faster and more stable convergence.

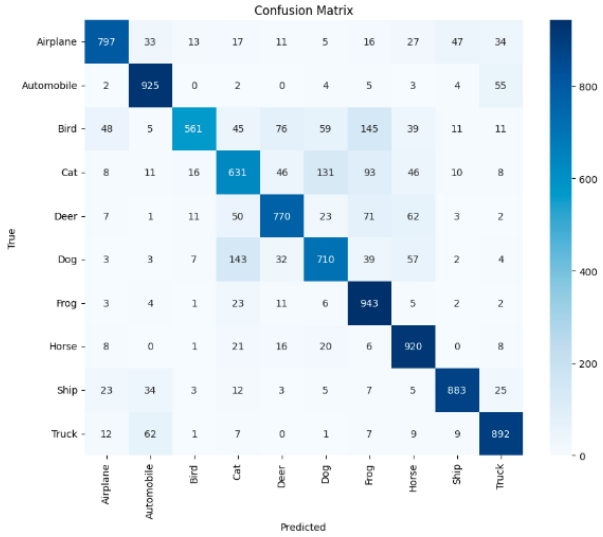


***Figure 11: Confusion matrix for AlexNet***

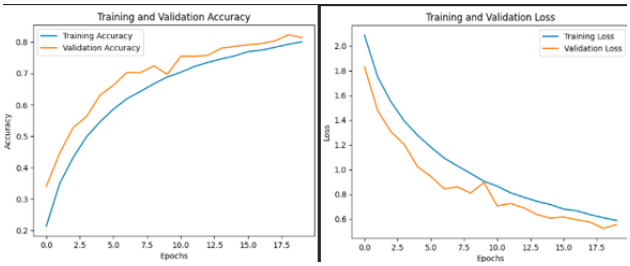


***Figure 12: Training accuracy vs validation accuracy and validation loss for AlexNet***

## Shortcomings and Room for improvement

The results on the data suggest that there is a lot of room for improvement.

### Resnet-18

Initial model was overfitting on the CIFAR-10 dataset. The training accuracy was high while the validation accuracy was on the lower end. This indicates poor generalization.

After improving the model by using L2 Regularization, data augmentation and mainly stochastic gradient descent (SGD) with momentum as the optimizer, the model started performing well. Different learning rates were also employed in the model which helped stabilize the model training and improve its validation accuracy.

The residual connections of the ResNet-18 model are one of the strengths of the model which helped avoid the vanishing gradient problem.

The shortcomings of this model are that it is still prone to overfitting without proper regularization and tuning. It requires larger epochs to converge into a solution.

The learning rate scheduler to adjust learning rates while training can be implemented in this model for further improvement of the model.

### EfficientNetB0

Although EfficientNetB0 is a model which is known to produce efficient results with high accuracy, it underperformed in this project. It produced the lowest accuracy compared to ResNet-18 and AlexNet models. The reason is because it is ineffective on smaller datasets like CIFAR-10. To

The EfficientNetB0 is tailored for larger datasets with higher resolution like 224*224 and performs bad on CIFAR-10 which has images with 32*32 resolution.

Resizing the images to a higher resolution like 224*224 can help the model perform better. The other way is to use lighter models of EfficientNetB0 like EfficientNet-lite to get better results.

### AlexNet

AlexNet is the best performer in our project for the CIFAR-10 dataset. It outperformed both the models because the model is designed for small datasets and resolutions like the CIFAR-10. The smaller depth of the AlexNet model helped avoid overfitting problems.

While the AlexNet model is efficient for small dataset, it would not perform well when subjected to larger datasets.

The model can be introduced to batch normalization layers to improve training convergence. Depth modifications can also go a long way in improving the model.

## Conclusion

In this project we explored three different models on the CIFAR-10 dataset. Comparing the three models we can summarize that the model architecture affects the model performance especially when datasets with specific resolutions and class distributions is applied.

While models like EfficientNetB0 and ResNet-18 can take complex features, smaller networks like AlexNet need simpler features and smaller datasets.

The major outline of this project refers to the fact that choosing the right model paves way to better accuracy and performance. The best model can be chosen taking into

account the size of the dataset, computational resources and the nature of the dataset.

The CNNs are the backbone of visual data classification and the right model with the right architecture is required for good accuracies and precisions.

## Code

The code for this project is available on https://github.com/tanvijois/Assignment2_DL.git [6]

## References


[1] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00444-8.

[2] H. C. Chen *et al.*, "AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf," *Electronics (Switzerland)*, vol. 11, no. 6, Mar. 2022, doi: 10.3390/electronics11060951.

[3] C. Shorten, T. M. Khoshgoftaar, and B. Furht, "Text Data Augmentation for Deep Learning," *J Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00492-0.

[4] D. Sarwinda, R. H. Paradisa, A. Bustamam, and P. Anggia, "Deep Learning in Image Classification using Residual Network (ResNet) Variants for Detection of Colorectal Cancer," in *Procedia Computer Science*, Elsevier B.V., 2021, pp. 423–431. doi: 10.1016/j.procs.2021.01.025.

[5] K. Kansal, T. B. Chandra, and A. Singh, "ResNet-50 vs. EfficientNet-B0: Multi-Centric Classification of Various Lung Abnormalities Using Deep Learning 'session id: ICMLDsE.004,'" in *Procedia Computer Science*, Elsevier B.V., 2024, pp. 70–80. doi: 10.1016/j.procs.2024.04.007.

[6] ChatGPT, https://chatgpt.com/. For a few code snippets