Experimentation with classical computer vision techniques inside a convolutional neural network

Ethan Leet
School of Information and Technology
Griffith University, Gold Coast campus
Australia
ethan.leet@griffithuni.edu.au

Abstract—**Convolutional Neural Networks (CNN) have been a forefront in computer vision research throughout the 20th and 21st centuries. Dominance in this field has been lead by big tech with resources that are out of reach for the every day academic. This research aims to provide insight into the CNN model with a focus on how classical computer vision techniques can improve a baseline CNN. Improvement in this context is defined as a decrease in time and processing power without a decrease in overall classification. This research starts by creating a baseline CNN model on a popular colour image dataset. Different image techniques ranging from blurring to edge detectors are then applied to the dataset and passed to the baseline CNN model. A further aim of this research is to experiment with multiple techniques and assess their validity in this type of image classification. Many of the techniques tested proved inefficient and irrelevant for this type of image classification. However, techniques such as blurring and grey scale conversions proved comparable to the baseline model.**

Keywords—CNN; LeNet-5; ReLU; max pooling; Softmax; Adam optimisation; cross entropy; 2D moving average; 2D accumulator

I. Introduction

Image classification is a prominent research area in the field of machine vision. Many different classification approaches exist with a convolutional neural network being a solid foundation for other models to build upon. The problem with extensive models is the amount of processing and man power required to build and run these systems.

The dataset used in this research is CIFAR-10, a 10 class colour image set consisting of 60,000 images[1]. The current leaderboard for this dataset consists of many different types of models and approaches. The highest CNN model on this leaderboard uses 121M parameters and achieves an accuracy of 99.1%[2]. It is evident that the leaders for this particular dataset (and many other datasets) have the capability to run these large networks using hardware not available to the everyday academic.

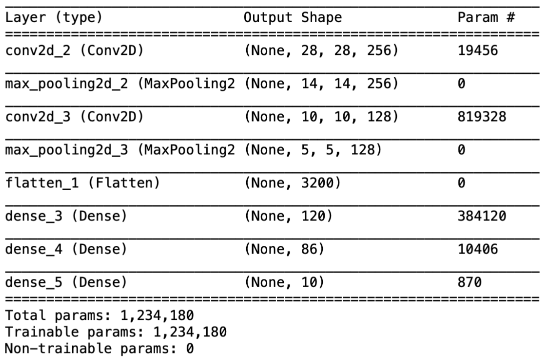
This research aims to provide an alternative approach for researchers in the field of machine vision through experimentation with different image techniques. Grayscale conversion, Box-Filtering, Gaussian-Smoothing, Canny Edge Detection, Harris Corner Detection and Hough-Line Transformation have been applied to the CIFAR-10 dataset with the aim to provide an increase in accuracy without increasing parameter size of the CNN model nor greatly increase the time required to train the model.

II. Previous Works

Many previous works have encapsulated the idea of reducing computational power of CNN models[3]. Further research has also been carried out with experiments of image techniques within other deep learning models as well as within CNN models[4][5][6]. [4][5] works have initialised the idea that classical machine vision techniques can prove effective for image classification. [6] proves beneficial to this research for a more in depth analysis into edge detection within a dataset and the resulting effect within a CNN model.

III. Technical Approach

In order to carry out experimentation with classical image techniques a baseline CNN model needed to first be created. Research into the LeNet-5 model proved a good starting point for a baseline model and an adapted LeNet algorithm was chosen as the baseline[7]. Fig 1 outlines the breakdown of the baseline model.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 28, 28, 256)       19456
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 128)       819328
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_3 (Dense)              (None, 120)               384120
_________________________________________________________________
dense_4 (Dense)              (None, 86)                10406
_________________________________________________________________
dense_5 (Dense)              (None, 10)                870
=================================================================
Total params: 1,234,180
Trainable params: 1,234,180
Non-trainable params: 0
```

Fig. 1. Outline of the models parameters

Fig. 1 consists of seven layers comprising; a convolutional layer with ReLU activation, a max pooling layer, a second convolutional layer, a second max pooling layer, followed by two fully connected layers with ReLU activation and finally an output layer with softmax activation[8][9].

To compile the model on the CIFAR-10 dataset the Adam optimisation technique was used with sparse cross categorical cross entropy as the accompanied loss function[10][11]. The model was then fitted and evaluated across 10 epochs with a full pass occurring on each epoch.

A visual example of the baseline model dataset can be found in Fig. 2.

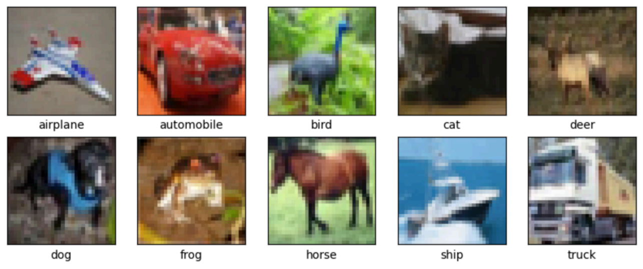


Fig. 2. CIFAR-10 dataset

A. Gray Scale

The first vision technique involved converting the whole dataset to a grayscale model. This was achieved by converting each image to 8bits and saving them to an array with one colour channel rather than three (RGB). The exact same parameter settings were used as outlined in Figure 1. This grayscale dataset was then fed through the network and saved for further image technique testing.

A visual example of the grayscale dataset can be found in Fig. 3.

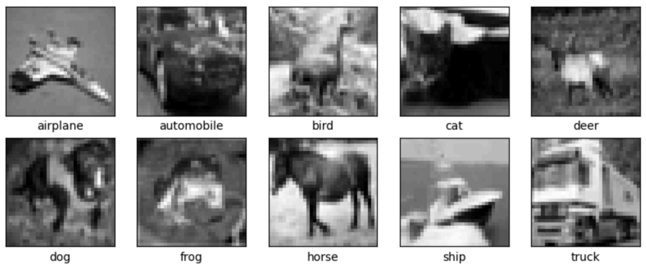


Fig. 3. CIFAR-10 dataset converted to 8bit grayscale

B. Box Filtering

Box filtering involves creating a kernel, which can be visualised as a NxN matrix, and moving it across an image one pixel at a time. Each pixel the kernel captures is averaged across the size of the kernel and the resulting pixel value is added to a new image. A visual representation of the kernel can be seen in Fig. 4.

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Fig. 4. A 3x3 normalised box filter

Box filtering across an entire image is calculated via the 2D moving average as defined below:

$$\begin{aligned} g[n,m] &= \frac{1}{9}\sum_{k=n-1}^{n+1}\sum_{l=m-1}^{m+1} f[k,l] \\ &= \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[n-k,m-l] \end{aligned} \tag{1}$$

A range of different kernel sizes were tested within the CNN model, a 5x5 kernel produced the better results out of all other kernel sizes tested. A visual representation of the dataset after the 5x5 Box Filtering technique was applied can be seen in Fig. 5.

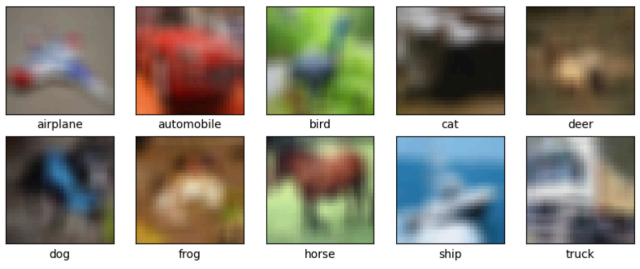


Fig. 5. CIFAR-10 dataset after box filtering

C. Gaussian Smoothing

The Gaussian filter operates on an image under a kernel size. The kernel is a NxN matrix where the real value of each pixel is captured. The resulting output pixel is then averaged across the sum of the matrix where neighbouring pixels have the most influence on the output pixel. An example filter can be found in Fig. 6.

$$K = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Fig. 6. 3x3 Gaussian Filter

The application of Gaussian smoothing reduces the image's high-frequency components and acts as a low pass filter for the image. [12] laid out work for the Gaussian Smoothing technique and is conducted by convolving an image with a Gaussian function:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

In this formula, x is the distance from the origin pixel in the horizontal axis, y is the distance from the origin pixel in the vertical axis and σ is the standard deviation of the Gaussian distribution.

A 5x5 Gaussian filter was chosen and its effect on the dataset can be visualised in Fig. 7.

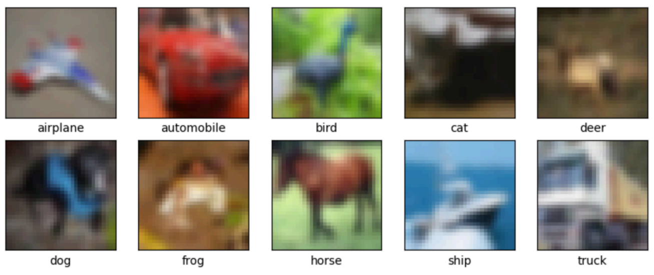


Fig. 7. CIFAR-10 dataset with 5x5 Gaussian Smoothing

D. *Canny Edge Detection*

John F. Canny in his works [13] created a popular and widely used edge detection algorithm known as Canny Edge Detection. The algorithm is multi-stage and can be broken down into (1) noise reduction, (2) finding intensity gradient of the image, (3) non-maximum suppression and (4) hysteresis thresholding.

1. *Noise Reduction*

As Canny laid out in [13], edge detection is susceptible to noise in the image. Firstly noise must be removed by applying a 5x5 Gaussian filter.

2. *Finding Intensity Gradient of the Image*

The first derivative in both the vertical (g_y) and horizontal (g_x) direction is then calculated on the smoothened image using the Sobel kernel[14]. Edge gradient magnitude and direction can then be calculated:

$$\text{Gradient} = \sqrt{g_y^2 + g_x^2} \quad \text{Direction} = \theta = \tan^{-1}\left[\frac{g_y}{g_x}\right] \quad (3)$$

3. *Non-maximum Suppression*

Non-maximum suppression involves removing any false pixels that may not be considered an edge. Each pixel is checked if it is at a local maximum in its neighbourhood of pixels in the direction of the gradient. Consider Fig. 8.

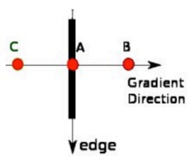


Fig. 8. Example of suppressing non-maximum edges

Here, point A is an edge where the gradient direction is normal to the edge. Point A is checked with points B and C as B and C are both in the gradient direction. If a local maximum is formed point A moves to the next step in the algorithm, otherwise point A is suppressed to zero.

4. *Hysteresis Thresholding*

This stage focuses on filtering weak edges from strong edges by removing small pixel noises on the assumption that edges are long lines and by discarding any edges out of a specific threshold. A minimum threshold value and maximum threshold value are used, where any detected edges over the maximum value are classified as strong edges and any detected edges below the minimum value are classified as weak edges. The last step in this thresholding is to classify all the detected edges between the minimum and maximum threshold values. If an edge between this threshold is connected to a strong edge it is classified as a strong edge, otherwise they are discarded. All strong edges are then returned by this stage of the algorithm.

Canny Edge Detection for this research was conducted on the grayscale dataset aforementioned. A minimum threshold value of 255/2 and a maximum threshold value of 255 were chosen. A visual example of the dataset after Canny Edge detection has been performed can be visualised in Fig. 9.

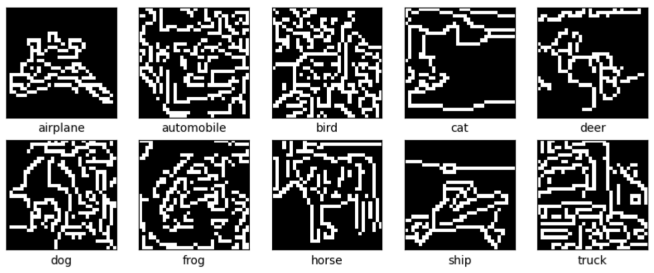


Fig. 9. CIFAR-10 dataset with 255/2 / 255 Canny Edge Detection

E. *Probabilistic Hough Line Transform*

In his works [15], Hough stipulated that the set of all lines going through a given point corresponds to a sinusoidal curve in the (p, θ) plane, and that two or more points on a straight line will give rise to sinusoids intersecting at the point for that line. He further states that any shape that can be represented mathematically, can be detected by the formula below, where p is the perpendicular distance from the origin of the line and θ is the angle formed by this perpendicular line.

$$p = x \cos\theta + y \sin\theta \quad (4)$$

To compute the Hough Transform we firstly must compute the lines for a given image, in this research, the Canny Edge method was used. Next, we can start the Hough Transform algorithm by firstly creating a 2D accumulator initialised to 0, where the column size of the accumulator is dependent on the desired accuracy (for one degree you will need 180 columns in the accumulator), and the rows (p) is the length of the image. Consider a horizontal line in an image, and take the first point (x, y) for that line. Now compute θ = 0, 1, 2, …., 180 and record the p value you get. For every (p, θ) pair, increment the value corresponding to the to the (p, θ) cells in the accumulator by one. Repeat this step for all points across the line, incrementing values in the accumulator.

This concept is considered voting for the (p, θ) values, at the end the (p, θ) cell with the highest vote will state that there is a line in this image of distance p from origin and at angle θ.

This algorithm is quite computationally expensive to run, the probabilistic Hough Transform is an optimisation of the aforementioned Hough Transform algorithm. It does not take all the points into consideration, instead it takes a random subset of points which is sufficient for line detection and uses a lower threshold value. A lowered thresholding value of π/180 and 50 was chosen. An example of Hough Line Transformation can be visualised in Fig. 10.

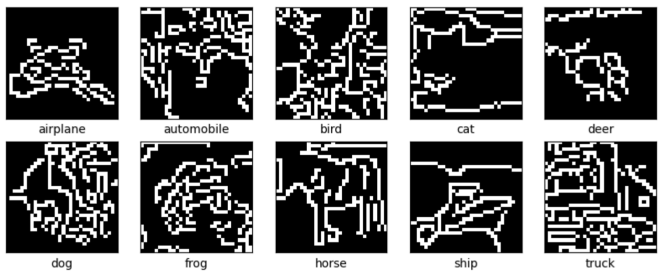


Fig. 10. CIFAR-10 dataset with π/180 / 50 Probabilistic Hough Line Transform

F. Harris Corner Detection

Corners are regions in the image with large variation in intensity in all directions. Harris and Stephens in their works [16] created a mathematical formula to represent the difference in intensity for a displacement of (u, v) in every direction. This can be expressed mathematically as follows:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2 \tag{5}$$

Where w(x, y) is the window function which is either rectangular or a Gaussian window, I(x + u, y + v) is the shifted intensity and I(x, y) is the intensity.

In order to maximise corner detection, the shifted intensity needs to be maximised, this can be conducted by applying the Taylor expansion which results in the following final equation:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{6}$$

Where:

$$A = \sum_{x,y} w(x,y) I_x^2 \qquad M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$
$$B = \sum_{x,y} w(x,y) I_x I_y \tag{7}$$
$$C = \sum_{x,y} w(x,y) I_y^2 \qquad = \sum_{x,y} w(x,y) \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix}$$

The result of this expansion will create an equation which will determine if a window can contain a corner or not:

$$\theta = \det(M) - \alpha \operatorname{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \tag{8}$$

In this formula, α should be considered constant and between a value of 0.04 and 0.06. Fig. 11 encapsulates a visual representation of the above formula, and whether it finds a corner, edge or a flat region.

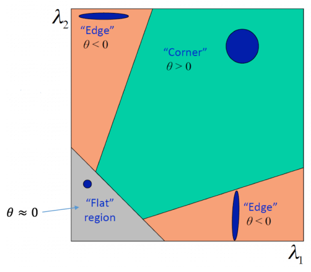


Fig. 11. Visual representation of the eigenvalues calculated and which region of the image they represent.

For this research, a α value of 0.04 was chosen. A visual representation of the dataset after Harris Corner detection can be seen in Fig. 12.

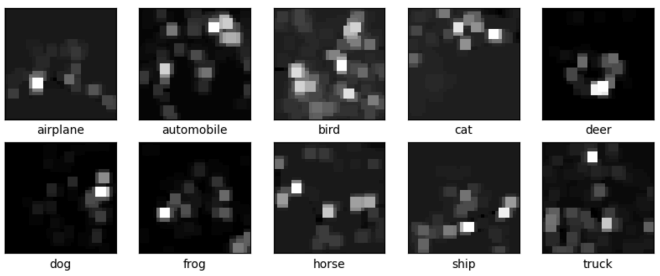


Fig. 12. CIFAR-10 dataset with α = 0.04 Harris Corner Detection

IV. EXPERIMENTS

1. *Baseline Model*

The baseline CNN model took 9.55 minutes to classify and reported an accuracy of 0.6672 and loss of 1.0202 for the validation set. Fig. 13 show a visual representation of the model.

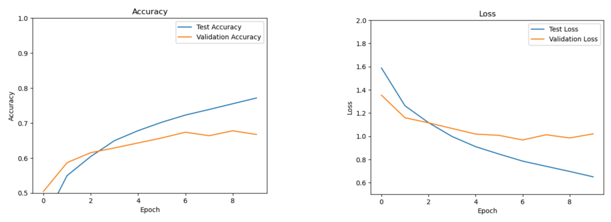


Fig. 13. CNN model accuracy and loss

2. *Grayscale Model*

The grayscale CNN model took 9.29 minutes to classify and reported an accuracy of 0.5831 and a loss of 1.4210 for the validation set. A visual representation of these results can be seen in Fig. 14.

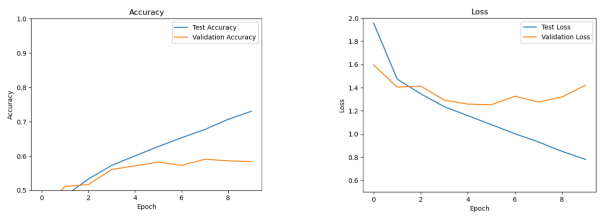


Fig. 14. CNN grayscale model accuracy and loss

3. *Box Filtering*

The Box Filtering technique took a total time of 9.62 minutes to classify and reported an accuracy of 0.5814 and loss of 1.2206 for the validation set. A visual representation of these results can be seen in Fig. 15.

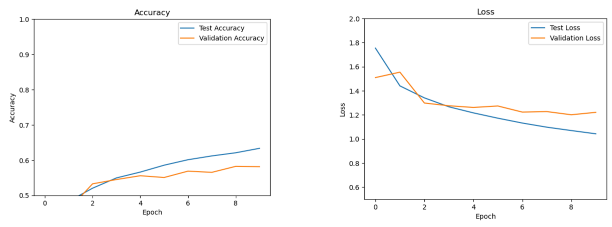


Fig. 15. Box Filtering technique accuracy and loss

4. *Gaussian Smoothing*

The Gaussian Smoothing technique took a total time of 9.74 minutes to classify and reported an accuracy of 0.6191 and a loss of 1.1339 for the validation set. A visual representation of these results can be seen in Fig. 16.

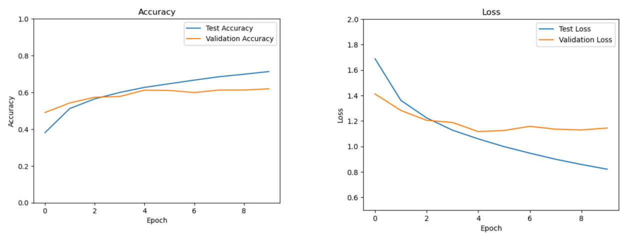


Fig. 16. Gaussian Smoothing technique accuracy and loss

5. *Canny Edge Detection*

The Canny Edge Detection technique took a total time of 8.68 minutes to classify and reported an accuracy of 0.5186 and a loss of 1.7515 for the validation set. A visual representation of these results can be seen in Fig. 17.

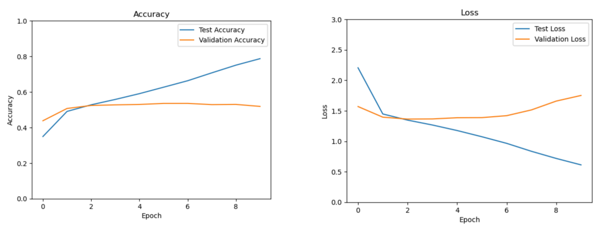


Fig. 17. Canny Edge Detection technique accuracy and loss

6. *Probabilistic Hough Line Transformation*

The Probabilistic Hough Line Transformation technique took a total time of 9.04 minutes to classify and reported an accuracy of 0.4972 and a loss of 2.0442 for the validation set. A visual representation of these results can be seen in Fig.18.

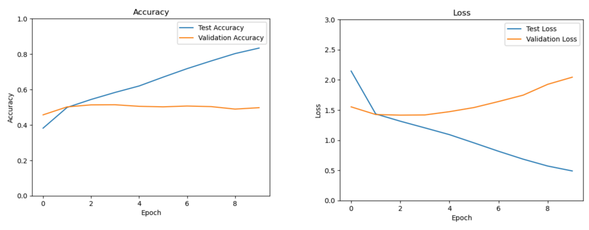


Fig. 18. Probabilistic Hough Line Transformation accuracy and loss

7. *Harris Corner Detector*

The Harris Corner Detection technique took a total time of 8.79 minutes to classify and reported an accuracy of 0.10 and a loss of 2.3027 for the validation set. A visual representation of these results can be seen in Fig. 19.

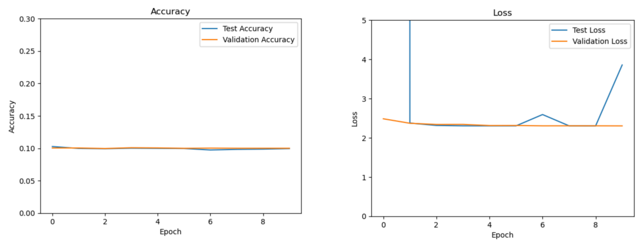


Fig. 19. Harris Corner Detection technique accuracy and loss

V. Conclusions

The experimental results obtained were not surprising for this type of research on this dataset. It can be seen that no classical image technique achieved a result comparable to that of an adapted LeNet-5 CNN model. The blurring techniques achieved a better result than the line and corner detection techniques. This is due to the amount of local image descriptors that are lost during the line and corner detection process. Although the times were slightly better for the model to run the manipulated datasets, the reduced accuracy does not warrant the necessity to use them. The Gaussian Smoothing technique was the closest technique to the baseline model, achieving an accuracy of 61.9% and a loss just slightly higher than the baseline model. Unlike the baseline model, the Gaussian Smoothing validation set tended closer to the test set accuracy as the model compiled and trained, outlining that this technique generalises the dataset well compared to the original dataset. It would be beneficial to further investigate the Gaussian Smoothing technique for this type of image classification.

References

1. "CIFAR-10 and CIFAR-100 datasets", *Cs.toronto.edu*, 2021. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed: 06- Oct- 2021].
2. "Papers with Code - CIFAR-10 Benchmark (Image Classification)", *Paperswithcode.com*, 2021. [Online]. Available: https://paperswithcode.com/sota/image-classification-on-cifar-10?tag_filter=3%2C5%2C4%2C14%2C6%2C0. [Accessed: 06- Oct- 2021].
3. M. Ahmadi, S. Vakili, J. Langlois and W. Gross, "Power Reduction in CNN Pooling Layers with a Preliminary Partial Computation Strategy", *2018 16th IEEE International New Circuits and Systems Conference (NEWCAS)*, 2018. Available: 10.1109/newcas.2018.8585433 [Accessed 6 October 2021].
4. R. Wang, W. Li, R. Qin and J. Wu, "Blur image classification based on deep learning", *2017 IEEE International Conference on Imaging Systems and Techniques (IST)*, 2017. Available: 10.1109/ist.2017.8261503 [Accessed 6 October 2021].
5. P. Shui and W. Zhang, "Corner Detection and Classification Using Anisotropic Directional Derivative Representations", *IEEE Transactions on Image Processing*, vol. 22, no. 8, pp. 3204-3218, 2013. Available: 10.1109/tip.2013.2259834 [Accessed 6 October 2021].
6. C. Xue, J. Zhang, J. Xing, Y. Lei and Y. Sun, "Research on Edge Detection Operator of a Convolutional Neural Network", *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2019. Available: 10.1109/itaic.2019.8785855 [Accessed 6 October 2021].
7. Y. LeCun et al., "Backpropagation Applied to Handwritten Zip Code Recognition", *Neural Computation*, vol. 1, no. 4, pp. 541-551, 1989. Available: 10.1162/neco.1989.1.4.541 [Accessed 6 October 2021].
8. K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biological Cybernetics*, vol. 36, no. 4, pp. 193-202, 1980. Available: 10.1007/bf00344251.
9. B. Gao and L. Pavel, "On the Properties of the Softmax Function with Application in Game Theory and Reinforcement Learning", *arXiv.org*, 2021. [Online]. Available: https://arxiv.org/abs/1704.00805. [Accessed: 06- Oct- 2021].
10. D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", *arXiv.org*, 2021. [Online]. Available: https://arxiv.org/abs/1412.6980v3. [Accessed: 06- Oct- 2021].
11. C. Robert, "Machine Learning, a Probabilistic Perspective", *CHANCE*, vol. 27, no. 2, pp. 62-63, 2014. Available: 10.1080/09332480.2014.914768.
12. D. House et al., "Computer Vision", *Computer Science Handbook, Second Edition CD-ROM*, 2004. Available: 10.1201/9780203494455.sec4 [Accessed 6 October 2021].
13. J. Canny, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. -8, no. 6, pp. 679-698, 1986. Available: 10.1109/tpami.1986.4767851 [Accessed 7 October 2021].
14. "(PDF) An Isotropic 3x3 Image Gradient Operator", *ResearchGate*, 2021. [Online]. Available: https://www.researchgate.net/publication/239398674_An_Isotropic_3x3_Image_Gradient_Operator. [Accessed: 07- Oct- 2021].
15. "INSPIRE", *Inspirehep.net*, 2021. [Online]. Available: https://inspirehep.net/literature/919922. [Accessed: 07- Oct- 2021].
16. *Bmva.org*, 2021. [Online]. Available: http://www.bmva.org/bmvc/1988/avc-88-023.pdf. [Accessed: 08- Oct- 2021].