Photo Classification using CIFAR-10

Rushabh Barbhaya

Satvika Shetty

Suresh Danala

Sakshi Nevatia

Harrisburg University of Science and Technology, Harrisburg PA

ANLY 535 – Machine Learning II

Dr. Roozbeh Sadeghian

Author Note

This paper is purely for academic use, do not use this documentation for research or other professional services. This document is articulated for the final project to the master's course of ANLY 535 – Machine Learning II for Harrisburg University of Science and Technology.

Abstract

When classifying photographs, it is necessary to build features capable of identifying patterns in images that reveal the identity of a group. The goal of this project was to categorize public CIFAR10 photos using deep learning image feature inputs. Convolutional neural networks (CNNs) all employ the same basic building block structure: convolutional layer modification and max-pooling, linked by a series of entirely interconnected layers. This research project at the usage of CIFAR-10 in photo classification. Data from the Canadian Institute for Advanced Research (CIFAR-10) is used to build an Artificial Neural Network (ANN) algorithm. The modified model attained an 88 percent classification accuracy rate after 10 hours. Keras, a Python tool, builds deep learning models.

Keywords

Jupyter notebooks, Keras, python, convolutional neural networks, Cifar-10, image categorization, CNN, UpSkilling, Conv2D, MaxPooling, De-Noise

Introduction

Since Yann LeCun's work with handwritten digits and Alex Krizhevsky's (AlexNet) obtained record performance metrics on the CIFAR dataset, convolutional neural networks (CNN) are used in image classification tasks has increased. Another critical benefit of CNNs is their ability to extract higher-level visual representations without the need for feature extraction, which is a time-consuming and expensive method that relies on domain knowledge to provide features for machine learning algorithms (Recht et al., 2018). Learnable filters are used in CNN to concatenate pictures at predefined intervals. CNN may enhance outcomes while employing fewer network parameters and lessening the computing burden. It takes much practice to be able to classify things appropriately. Many people make errors before they do it right the first time. The structure of machine learning is similar. Deep learning can categorize items better than humans when given a high-quality data set. People may be able to dedicate more time to more enjoyable pursuits by automating some of the more repetitive chores if a highly accurate picture classifier is developed.

Because some of the traits that constitute a class are barely visible to human eyes, achieving a high classification rate using a series of microscopic images is challenging. Computational vision is a subject that is constantly evolving to understand and precisely identify each item. Autonomous vehicles, for example, may benefit from this sort of study since they regularly fail to distinguish objects in specific settings, resulting in significant harm. For the great majority of available problems, the vast majority of traditional neural network algorithms do not generate good results (Calik et al., 2018). Because of this knowledge, automated jobs cannot be substituted. This work uses a unique CNN structure to categorize pictures from the CIFAR-10 dataset into ten separate groups based on geographical distribution. A two-dimensional convolution layer is utilized instead of the original model's max-pooling and dense function to improve classification rate and accuracy.

Background Information

Back-propagation boosts the network's accuracy during training in conventional convolutional neural networks, which initialize all of the network layers' weights simultaneously. As network depth rises, so does computational expense, and test accuracy suffers as a result (Zhu et al., 2020). Therefore, an incremental method for reinitializing the weights of each layer, in which the importance of the previous layer is determined and left unchanged for a predetermined period, followed by a step in which the weights of all subsequent layers are initialized, and this process is repeated until the consequences of all layers are established (Chauhan et al., 2018). Using the CIFAR10 dataset, several investigations were carried out to assess the method's performance. According to the study's results, this strategy is nine percent more accurate and 29 percent less time-consuming.

This method allows the network to be trained faster and more precisely. Before training deep learning models, it is vital to understand the parameters needed to get the desired result. The overall goal is to minimize time and money spent on education. A model is fed data samples and results in supervised learning (Doon et al., 2018). Models create output by comparing it to the intended outcome and then producing it as near the desired output as possible. This is accomplished via the application of optimization techniques. The model’s accuracy improves as the optimization approach cycles through multiple iterations until convergence is attained. Several ways for overcoming barriers that arise throughout the learning process exist. SGD, rmsprop, adadelta, Adam, and adagrad are six of the most often used techniques.

Related work

Photo Classifications are commonly used in autoencoders by organizations. An autoencoder is a form of unsupervised artificial neural network that learns efficient data coding. An autoencoder's goal is to train the network to ignore signal "noise" to understand a representation (encoding) for a data set, generally for dimensionality reduction. Companies use autoencoders to get to know their customers well.

Let’s look at a simple credit-risk example and learn how to unearth new patterns, anomalies, and actionable insights from data. The autoencoder w learn the characteristics that distinguish a "good" credit risk from "bad" credit risk. Outliers and uncommon events, for example, will be removed from the data. The trained autoencoder will compress the data before rebuilding it to its original size using the compressed representation. Any consumer the model, has trouble reuniting with could be a red flag that must be investigated further.

Some of the other applications of Autoencoder are:

Anomaly Detection: An autoencoder that has been adequately trained can reconstruct the data input with the least amount of reconstruction error. When any outliner or anomaly is fed into a trained autoencoder, the output is drastically different from the input and contains a substantial error term, indicating an anomaly.

Image Super-Resolution: There are a variety of strategies for enhancing image resolution, including bicubic, bilinear, and others; however, they are all interpolation algorithms with restrictions.

Fraud Detection: Given how small fraudulent transaction counts are compared to the overall number of transactions in a company, training machine learning models to learn about fraudulent behavior can be difficult. Compared to older methods, autoencoders' adaptability allows users to generate data projections for modeling fraudulent transactions. Once trained, autoencoders can generate additional data points and create similar fraudulent transactions, providing a broader data set for machine learning models to learn. This is especially helpful in situations where we do not have enough historical samples of fraudulent transactions or when entirely new patterns of fraudulent transactions emerge.

Data

The CIFAR-10 dataset consists of 60,000 images in 10 classes; 6000 images per class. CIFAR is an acronym for Canadian Institute For Advanced Research. The ten classes/categories of the dataset are Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck. There are three versions of the dataset: python, MATLAB, and a binary version.

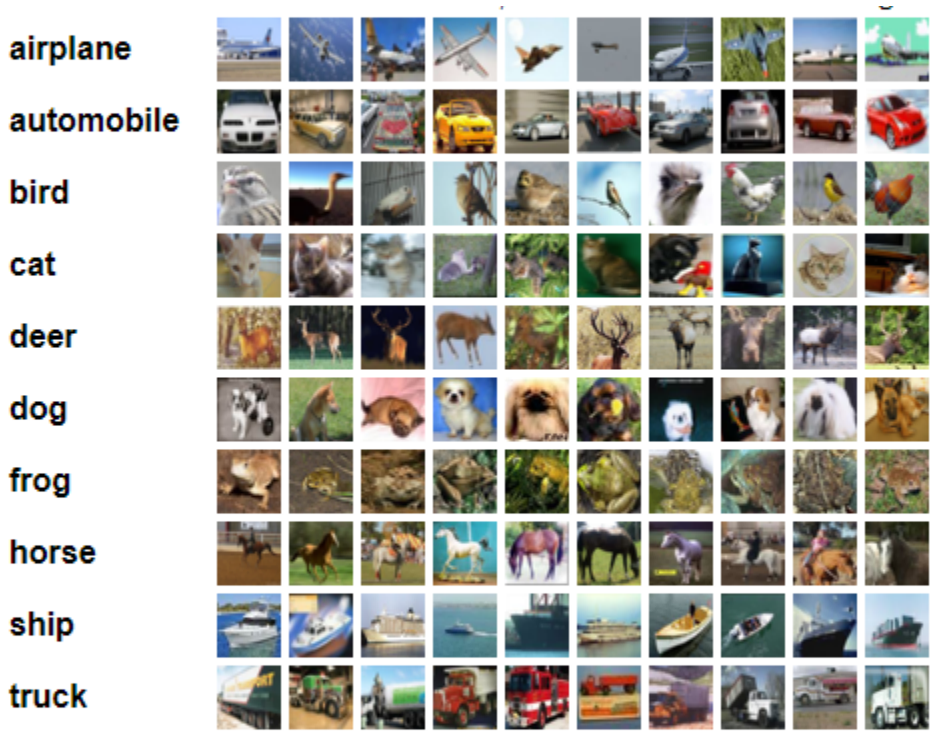

Figure 1: CIFAR-10 Sample Dataset

Technical approach

This research proposes two applications. One is denoising images, and the other is recognizing the images. These applications use Convolution Neural Network (CNN) with different supported layers to help with the application's accuracy and speed.

De-Noising Model

This application is to exercise one of the machine learning models' applications to re-create and remove the noise from the images. The dataset doesn't contain any noise on the image. The noise is added to the images using simple functions. The goal is to get the original image back after adding the noise to the pictures. The training set uses 50,000 photographs, and 10,000 images are used for testing. The actual size of this image is 32 x 32.

Four models were created to test different functionalities from the machine learning models. The first model used two convolutional dimension layers of 32 and 64 units with Batch Normalizer and Max Pooling. An Adam optimizer with a learning rate of 1/10,000. To stabilize the image, Up Sampling was used in the decoder section. Up Sampling increases the dimension, and at the end of the decoder, we get the exact image size and resolution as the input. Conv2dTranspose is utilized in the decoder to the Conv2d of the model for the other models. Up Sampling and Convolution are both performed on these models. For model 4, autoencoding is used to help get more apparent resolutions on the decoded images.

Object Recognition

The same dataset is used for object recognition. As stated earlier, there are 50,000 images for training and 10,000 images for testing. The preliminary analysis says that the model should be overfitted. The model would be heavily skewed as there are 6,000 images for each class. For this application, Conv2d and, MaxPool to get an accurate model. The patience model is used to stop the model when there's no improvement preemptively. A learning rate of 1/1000 is used.

Test and evaluation

De-Noise Models

Model 1

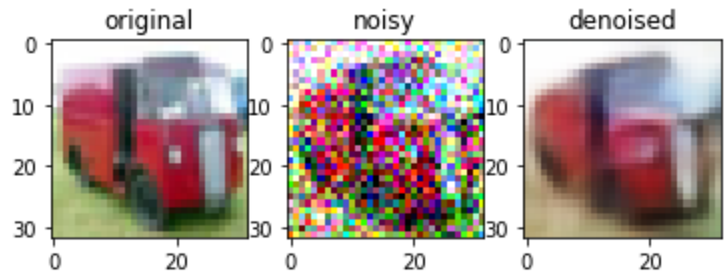

Figure 2: De-Noise Model 1

This model used to remove noise from the pictures successfully removes all the noise from the source images. However, the photos are not 100% clear and have some blurry lines.

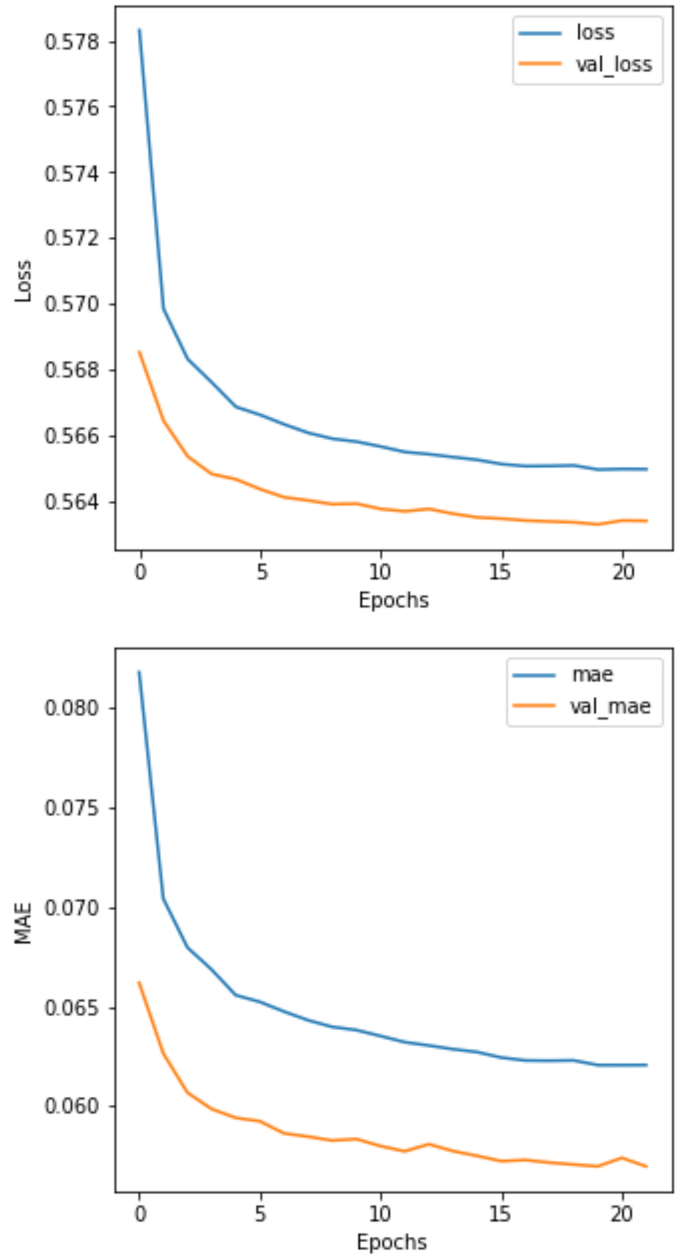

Figure 3: De-Noise Model 1 Learning curves

The model's training and validations models closely follow each other, showing a standard model curve, i.e., no underfit or overfit of the models. The model's accuracy is a respectable 72%

Model 2

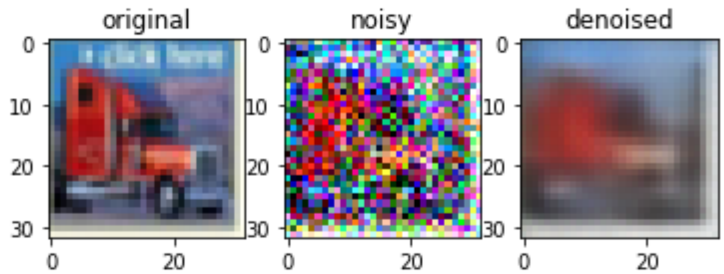

Figure 4: De-Noise Model 2

This model's output images are blurry and incomprehensible.

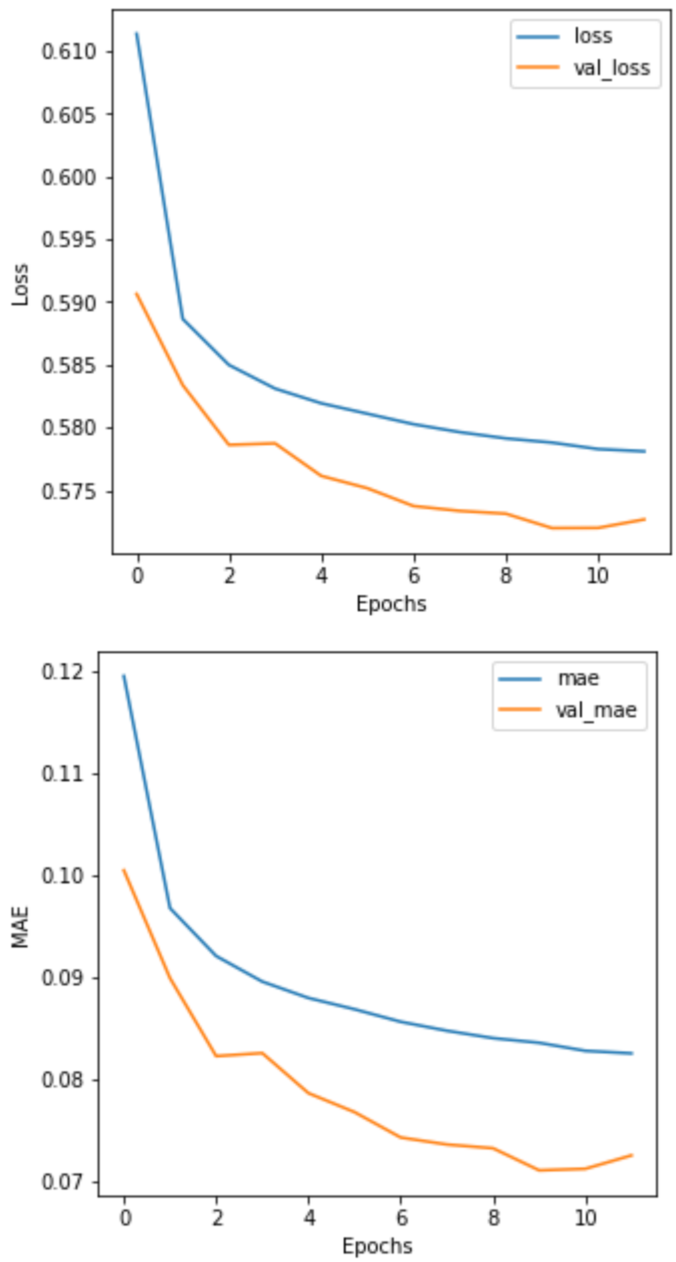

Figure 5: De-Noise Model 2 learning curves

The model's curves are close to each other, showing a normal response. This model offers a slight skew towards underfit. The model's accuracy is 69.34%

Model 3

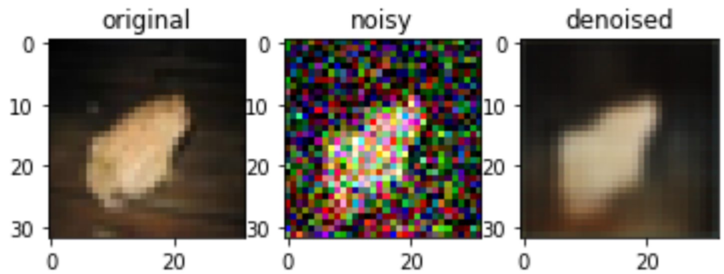

Figure 6: De-Noise Model 3

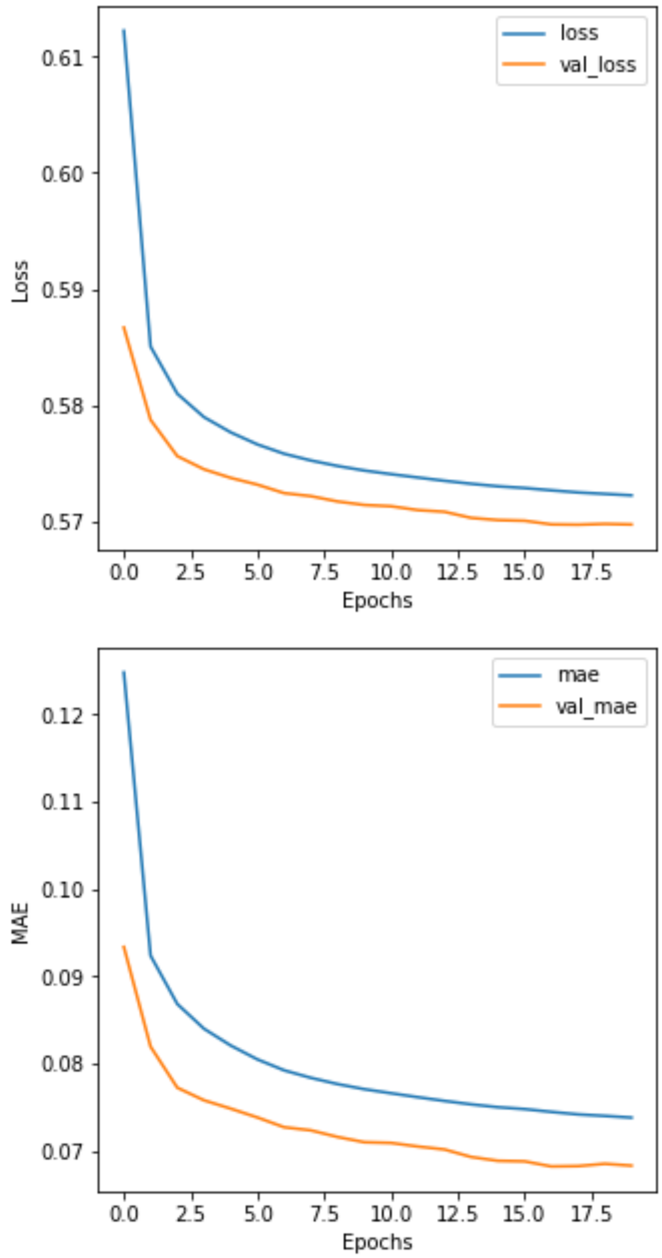

Figure 7: De-Noise Model 3 learning curves

This model has some definitions but is still unclear. The learning curves are closer as compared to model 2, with an accuracy of 65.5%

Model 4

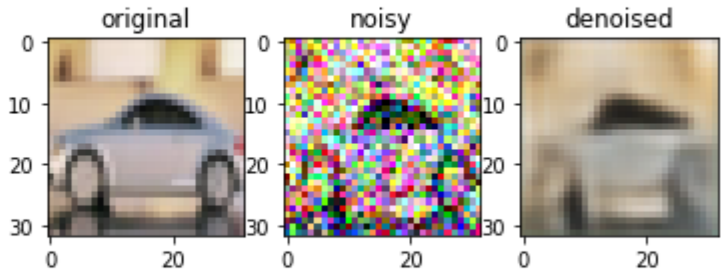

Figure 8: De-Noise Model 4

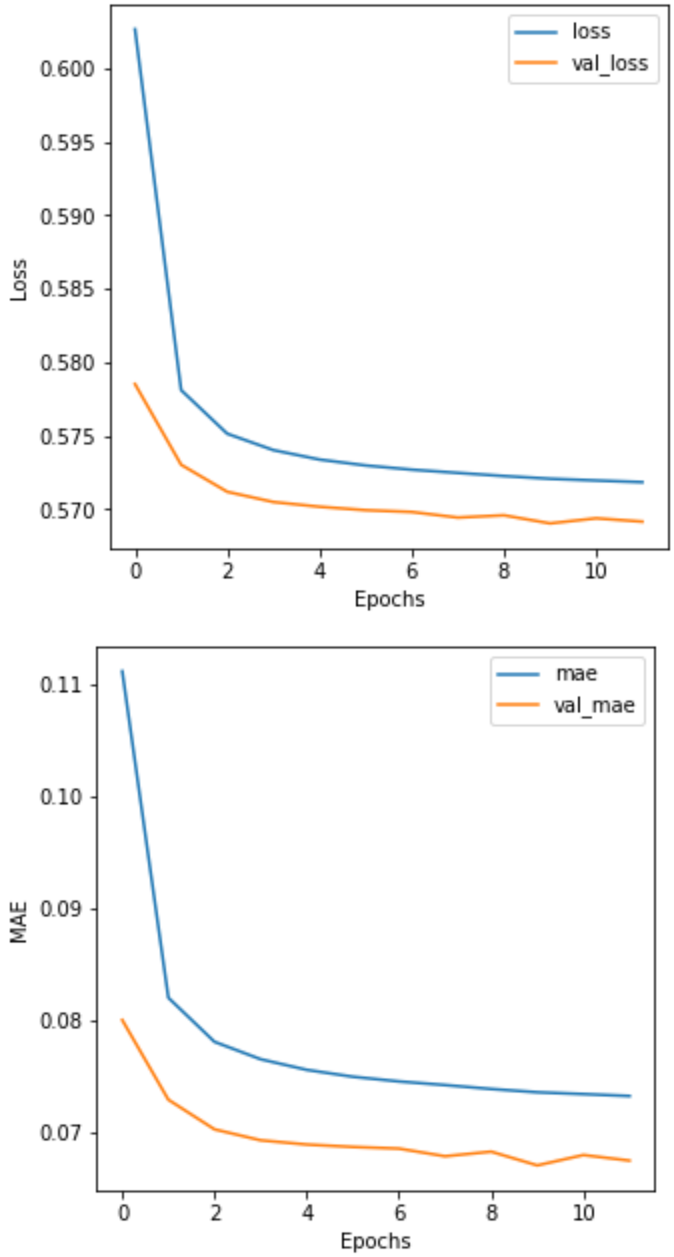

Figure 9: De-Noise Model 4 learning curves

Model 4 is similar to model 3. The model shows a structure of the image but not much definition in the output images. This model has an accuracy of 64.9%

Object Recognition Model

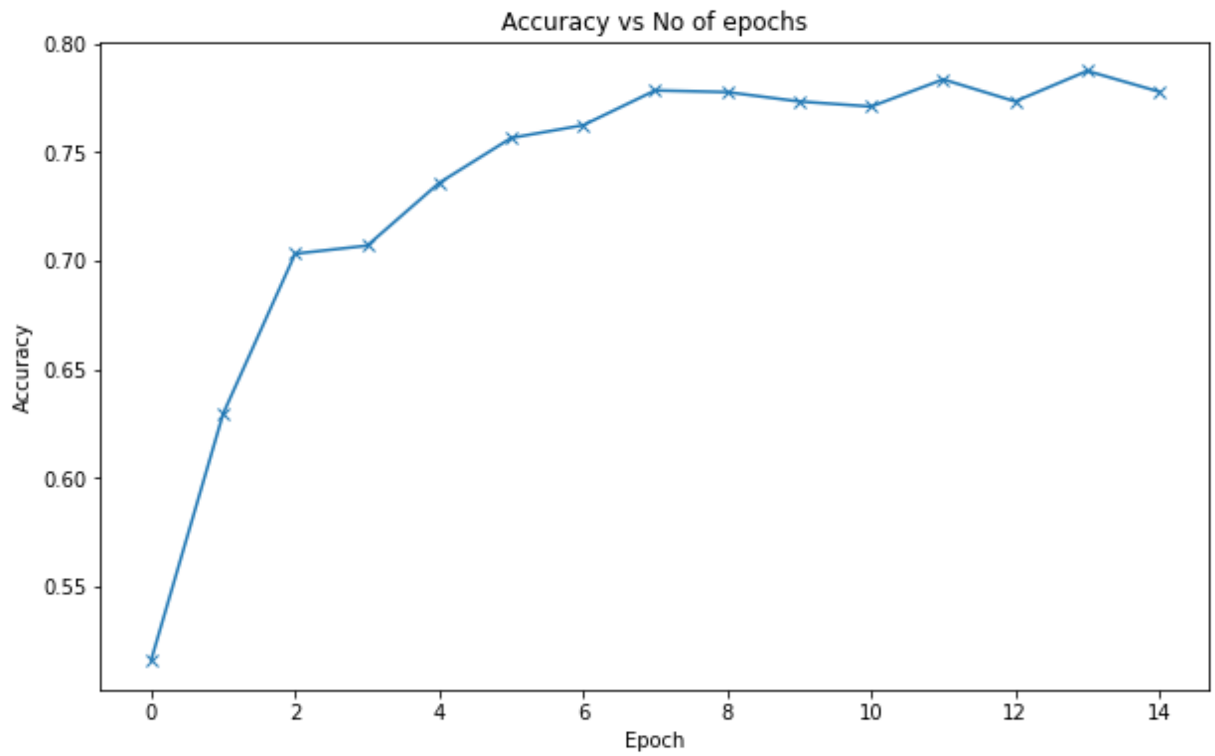

Figure 10: Object Recognition - Accuracy vs. Epoch curve

The object recognition model is the focus of this research. As the number of epochs increases, it reaches a stable stage where the number of epochs doesn't affect the accuracy.

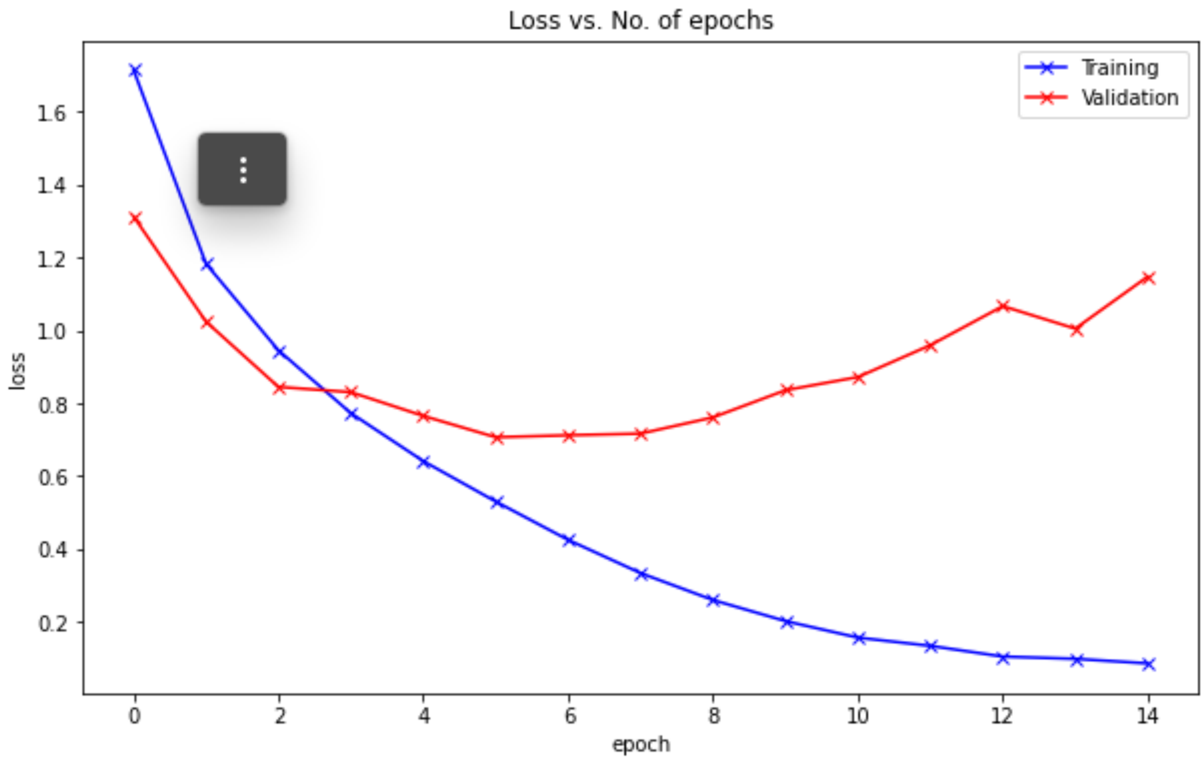

Figure 11: Object Recognition - Learning Curves

The model's learning curves show overfitting, as was suspected with the initial analysis. The loss coefficient of the validation curve is ever-increasing and doesn't seem to follow the training curve. The model's accuracy is close to 77.8% at the end of 14 cycles.

Conclusion

According to the findings of this study, it is possible to improve the accuracy of CNN models by using computer language that is quickly executed and modified. When comparing accuracy and runtime, it's worth noting that this model required more computing power to achieve its highest accuracy. Still, when it came to running time/accuracy, it won with the most common CNN structure. AI developers would have to decide whether the more powerful models are worth the extra effort based on the available components.

References

Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2018). Do CIFAR-10 classifiers generalize to CIFAR-10?. *arXiv preprint arXiv:1806.00451*.

Zhu, M., & Chen, Q. (2020). Big data image classification based on distributed deep representation learning model. *IEEE Access*, *8*, 133890-133904.

Çalik, R. C., & Demirci, M. F. (2018, October). Cifar-10 image classification with convolutional neural networks for embedded systems. In *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)* (pp. 1-2). IEEE.

Doon, R., Rawat, T. K., & Gautam, S. (2018, November). Cifar-10 classification using deep convolutional neural network. In *2018 IEEE Punecon* (pp. 1-5). IEEE.

Chauhan, R., Ghanshala, K. K., & Joshi, R. C. (2018, December). Convolutional neural network (CNN) for image detection and recognition. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)* (pp. 278-282). IEEE.