# Introduction

Introduction

In the realm of modern data science and machine learning, the analysis of massive datasets have become quintessential for obtaining insights and developing advanced predictive and classification models. The advent of big data technologies has ushered in a new era where the storage and processing of large data volumes is not only feasible but essential for solving complex problems across various domains. Within this context, the choice of an appropriate data storage system plays a pivotal role in enabling efficient data management, retrieval, and analysis. Alongside this, the rapid advancements in deep learning have led to the emergence of complex neural network architectures, including Convolutional Neural Networks (CNNs), which have demonstrated impressive ability in media classification, including classification of images.

This research paper looks to explore some big data storage solutions, with a particular focus on the Hadoop Distributed File System (HDFS). At the same time, it analyses the use of advanced neural network models, particularly CNNs equipped with sequencers, for the classification of a widely studied dataset - CIFAR-10. Through this analysis, it aims to provide valuable insights into the scalable data storage solutions as well as cutting-edge deep learning techniques for the efficient processing and analysis of large-scale image datasets.

Big Data Storage Options

In today's data-driven world, managing and storing vast datasets pose significant challenges. Traditional storage solutions often fall short in terms of scalability, fault-tolerance, and cost-effectiveness when dealing with big data. HDFS is a distributed file system designed explicitly to tackle these challenges. HDFS partitions large files into smaller blocks, distributes them across a cluster of machines, and replicates them to ensure fault tolerance. Its robust architecture makes it a popular choice for storing and managing large datasets, making it a core component of the Hadoop ecosystem.

In this paper, we delve into the core features of HDFS and examine its role in the efficient storage of large-scale datasets. We explore HDFS's fault-tolerant design, data replication mechanisms, and its integration with Hadoop's distributed processing framework, MapReduce. We also discuss its applicability to diverse use cases and its suitability for data-intensive applications, such as machine learning and deep learning tasks.

Deep Learning with CNN Sequencers

In recent years, deep learning has revolutionised the field of machine learning, especially in image-related tasks. While some argue that CNNs cannot handle very large images, they are still a very popular choice (Das et al., 2020), showcasing impressive capabilities in image classification, object detection, and image generation. Moreover, CNNs can be further enhanced with sequencers, allowing them to capture temporal dependencies in sequential data, opening doors to sophisticated tasks such as video analysis, speech recognition, and sequential image classification.

In this research, the design and implementation of CNN sequencers is investigated, along with their application to image classification tasks.

# Dataset

The CIFAR-10 dataset, short for the Canadian Institute For Advanced Research - 10, was created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton in 2009. It consists of 60,000 32x32 color images spread across 10 distinct classes, with each class containing 6,000 images. These classes are:

Airplane

Automobile

Bird

Cat

Deer

Dog

Frog

Horse

Ship

Truck

The dataset is divided into two subsets: a training set with 50,000 images and a testing set with 10,000 images, making it suitable for both training and evaluation purposes. One of the key characteristics of CIFAR-10 is that it contains real-world, low-resolution images, which poses several unique challenges and benefits for CNN-based research.

Advantages for CNN Perspective:

Small Image Size: The 32x32 pixel resolution of CIFAR-10 images is relatively small compared to many other datasets in computer vision. This small size makes it computationally efficient, allowing researchers to experiment with CNN architectures more quickly.

Diversity of Classes: CIFAR-10 encompasses a diverse range of object classes, including animals, vehicles, and more. This diversity encourages the development of models that can handle a wide array of visual concepts, making it valuable for general-purpose image classification tasks.

Real-World Complexity: The images in CIFAR-10 are more complex and varied compared to synthetic or highly curated datasets. This realism is crucial for training CNNs to work effectively in real-world scenarios, where images are often noisy, contain various backgrounds, and exhibit different lighting conditions.

Comparative Benchmark: CIFAR-10 has become a standard benchmark in the field of computer vision. Many state-of-the-art CNN architectures and techniques are evaluated on this dataset. This makes it easier to compare and benchmark the performance of new CNN models against existing ones, facilitating progress in the field.

Accessibility: CIFAR-10 is freely available for research purposes, making it accessible to a wide range of researchers and students. Its availability has contributed to its popularity in the machine learning community.

In conclusion, the CIFAR-10 dataset is a valuable resource for researchers working with CNNs. Its small image size, diverse classes, real-world complexity, limited data, and status as a benchmark dataset make it an excellent choice for exploring and advancing CNN-based computer vision research. Researchers often use CIFAR-10 as a stepping stone to develop and fine-tune their models before applying them to more challenging and extensive datasets, making it an integral part of the computer vision research landscape.

# Literature review

Different researchers have provided many solutions to the storage and processing of image datasets such as the Cifar10 dataset. One survey was carried out in 2016 **(Lim, Young and Patton, 2016),** which looked to employ the caffe deep learning framework to train neural network models against the well-known MINST, CIFAR10 and ImageNet. It could be assumed that one reason the researchers choose the Caffe framework is because it is a well-established deep learning framework and is relatively easy to use **(Majumder et al., 2017).** Lim, Young and Patton analysed the effectiveness of the following systems:

(1) PNG-formatted image files on local file system

(2) Pushing pixel arrays from image files into a single HDF5 file on local file system;

(3) In-memory arrays to hold the pixel arrays in Python and C++;

(4) Loading the training data into LevelDB, a log-structured merge tree based key-value storage; and

(5) Loading the training data into LMDB, a B+tree based keyvalue storage

The paper successfully highlights the inefficiencies of using image files that are saved locally on systems such as ext4 – the study notes that training can be up to 17 times slower. It could be regarded as unfortunate that the researchers didn’t extend the survey to encompass more efficient systems such as Amazon S3, Google Cloud Storage or HDFS which is being used for this research paper.

The paper also concludes that to effectively train deep neural networks, it is common to use large datasets that are accessed through mini-batches during training. However, traditional file systems are not well-suited for this task due to their inefficient indexing and caching mechanisms. Instead, they recommend utilizing image storage backends with efficient indexing capabilities for training samples and improved caching mechanisms. This approach will enhance the training process of deep neural networks by optimizing data access and retrieval.

**PHDFS: Optimizing I/O performance of HDFS in deep learning cloud computing platform (Zhu et al., 2020)**

This research paper discusses the significance of the file system in deep learning cloud computing platforms, highlighting the use of Hadoop Distributed File System (HDFS) in large-scale clusters due to its high performance and availability. However, it points out a critical issue in deep learning datasets where the number of files is extensive, but individual file sizes are small, leading to performance problems in HDFS.

To address this issue, the paper introduces a new approach called Pile-HDFS (PHDFS), which focuses on aggregating small files based on their correlation. The concept of "Pile" is introduced as an I/O unit that combines groups of small files with similar characteristics. Additionally, the research proposes a two-layer manager and incorporates internal organization information into data blocks to improve small file access efficiency.

The experimental results presented in the paper show that PHDFS outperforms the original HDFS significantly. It reduces latency when accessing small files and enhances the Frames Per Second (FPS) of typical deep learning models by up to 40%. This suggests that PHDFS offers a valuable solution to the performance challenges associated with deep learning datasets in cloud computing environments.

**Detecting Anomaly in Big Data System Logs Using Convolutional Neural Network (**(Lu et al., 2018)**)**

Some criticism of Hadoop: *For example, Hadoop and Spark applications*

*often demand long execution duration, thus a huge size of*

*logs will be generated [7], [8]. Furthermore, each system*

*may employ its own logging framework such as log4j [9]*

*and self4j [10]; hence log formats could be diverse. Moreover, some unexpected events happening during the program*

*execution might cause big performance degradation, or even*

*failures [11], [12], [13]. Those scenarios are hard to be*

*detected manually, even for system experts.*

**“Mrapid: An efficient short job optimizer on Hadoop (**(Zhang, Huang and Wang, 2017)**)**

**^This is a good paper with a solution to Hadoop’s inefficiency:**

*Abstract—Data have been generated and collected at an accelerating pace. Hadoop has made analyzing large scale data much simpler to developers/analysts using commodity hardware. Interestingly, it has been shown that most Hadoop jobs have small input size and do not run for long time. For example, higher level query languages, such as Hive and Pig, would handle a complex query by breaking it into smaller adhoc ones. Although Hadoop is designed for handling complex queries with large data sets, we found that it is highly inefficient to operate at small scale data, despite a new Uber mode was introduced specifically to handle jobs with small input size. In this paper, we propose an optimized Hadoop extension called MRapid, which significantly speeds up the execution of short jobs. It is completely backward compatible to Hadoop, and imposes negligible overhead. Our experiments on Microsoft Azure public cloud show that MRapid can improve performance by up to 88% compared to the original Hadoop.*

# Methodology

We are performing a data pipeline for deep learning with the CIFAR-10 dataset. First, we are saving the CIFAR-10 dataset into the Hadoop Distributed File System (HDFS), which is a distributed and scalable storage system. This step allows us to efficiently store and manage the dataset in a distributed computing environment.

Next, we are importing the CIFAR-10 dataset into Python using the Pickle module. Pickle is a Python library that allows us to serialize and deserialize Python objects, making it a convenient choice for loading and working with datasets.

Finally, we are running a neural network using the imported CIFAR-10 dataset. This neural network is designed for tasks such as image classification. The neural network will be trained on this data to learn patterns and features within the images, enabling it to make predictions or classifications based on new, unseen images.

In summary, our workflow involves storing the CIFAR-10 dataset in HDFS for efficient storage, importing it into Python using Pickle for data manipulation, and then training a neural network on this dataset for image-related tasks.

# Results and discussion

Title: Results and Discussion - Training a Neural Network on CIFAR-10 Dataset

Introduction:

In this section, we will discuss the results of training a neural network on the CIFAR-10 dataset using Python and Keras. The dataset was preprocessed, and a convolutional neural network (CNN) model was constructed. The model was trained for 150 epochs, and the achieved accuracy and loss will be analyzed and discussed in detail.

Results:

After training the CNN model on the CIFAR-10 dataset for 150 epochs, the following results were obtained:

1. Accuracy: 53%

2. Loss: Approximately 1.33

Discussion:

Now, let's delve into the discussion of these results, their implications, and potential areas for improvement.

1. \*\*Accuracy of 53%\*\*:

- The achieved accuracy of 53% indicates that the model was able to correctly classify approximately 53% of the images in the test dataset. While this accuracy is better than random guessing (10% for a 10-class problem), it may not be considered high for practical applications.

- Factors contributing to this accuracy include the complexity of the CIFAR-10 dataset, the architecture of the neural network, and the training parameters. CIFAR-10 consists of 60,000 32x32 color images across 10 classes, making it a challenging benchmark for image classification tasks.

- Potential areas for improvement in accuracy include experimenting with different architectures, hyperparameters, and regularization techniques. Increasing the depth of the CNN, adjusting learning rates, and implementing techniques like dropout or batch normalization can potentially enhance accuracy.

2. \*\*Loss of Approximately 1.33\*\*:

- The loss value of approximately 1.33 indicates the error or mismatch between the predicted and actual labels during training. Lower loss values are generally desirable, as they signify a better fit of the model to the training data.

- The loss value should be interpreted in the context of the problem and dataset. In some cases, a higher loss might be acceptable if it corresponds to a high level of class imbalance or noisy data.

- To reduce the loss further, one could explore different optimization algorithms, such as stochastic gradient descent with momentum (SGD) or adaptive learning rate methods like Adam. Additionally, fine-tuning the model's hyperparameters and using data augmentation can help in minimizing the loss.

3. \*\*Data Augmentation\*\*:

- Data augmentation was applied to the training images using techniques such as shifting, rotation, and flipping. This helps in increasing the diversity of the training data, which can lead to better generalization.

- Data augmentation is a valuable tool for improving model performance, especially when the dataset is limited in size. However, its impact on the final results may vary depending on the dataset and the problem. It's essential to strike a balance between augmentation and overfitting.

4. \*\*Model Architecture\*\*:

- The model architecture consisted of convolutional layers followed by max-pooling layers and fully connected layers. This architecture is a common choice for image classification tasks.

- Experimenting with different architectures, such as deeper networks or using pre-trained models like VGG, ResNet, or Inception, may yield better results. Additionally, fine-tuning the number of filters, kernel sizes, and the use of dropout layers can be explored.

5. \*\*Epochs and Training Duration\*\*:

- Training the model for 150 epochs suggests a prolonged training duration. It's essential to monitor the model's performance on validation data throughout training to avoid overfitting.

- Techniques such as early stopping can be implemented to halt training when validation performance plateaus or starts degrading. This can save computational resources and prevent overfitting.

6. \*\*Validation Split\*\*:

- A validation split of 20% was used to monitor the model's performance during training. This is a standard practice to ensure that the model generalizes well to unseen data.

- Hyperparameter tuning, such as the learning rate and batch size, can be performed based on validation performance.

Conclusion:

In summary, training a neural network on the CIFAR-10 dataset resulted in an accuracy of 53% and a loss of approximately 1.33 after 150 epochs. While these results demonstrate a certain level of success, there is room for improvement in both accuracy and loss. Experimentation with model architecture, hyperparameters, and regularization techniques can lead to better performance. Additionally, careful monitoring of validation metrics and early stopping can prevent overfitting during training. Ultimately, achieving higher accuracy and lower loss is desirable, especially for practical applications of image classification.

# Conclusion and Future Works

Title: Conclusion and Future Directions in Training a Neural Network on CIFAR-10

\*\*Conclusion\*\*:

In this research endeavor, we embarked on the challenging task of training a convolutional neural network (CNN) on the CIFAR-10 dataset. After extensive experimentation and training over 150 epochs, we achieved an accuracy of approximately 53% and a loss of about 1.33. While these results demonstrate a commendable effort, there are numerous avenues for future work and improvements in both model performance and research scope.

The accuracy of 53% suggests that our model has learned meaningful patterns within the CIFAR-10 dataset but still has room for enhancement. It's important to recognize that the CIFAR-10 dataset is notoriously intricate, containing 60,000 32x32 color images across 10 diverse classes. This inherent complexity makes it a formidable benchmark for image classification tasks. In the subsequent sections, we discuss potential future directions to elevate our research and its practical applications.

\*\*Future Directions\*\*:

1. \*\*Model Architecture Exploration\*\*:

- One of the primary future directions is to explore more intricate model architectures. While our current CNN architecture is effective to a certain extent, more sophisticated models such as ResNet, Inception, or EfficientNet have shown exceptional performance in image classification tasks. These architectures can be adopted and fine-tuned to potentially achieve significantly higher accuracy.

2. \*\*Hyperparameter Tuning\*\*:

- Delving into hyperparameter tuning is crucial for further improving model performance. This includes optimizing the learning rate, batch size, and weight initialization schemes. Exhaustive grid or random search over a range of hyperparameters can lead to the discovery of configurations that offer superior results.

3. \*\*Regularization Techniques\*\*:

- Incorporating regularization techniques like dropout, batch normalization, and L1/L2 regularization can mitigate overfitting and help the model generalize better. Fine-tuning the dropout rate and evaluating its impact on model performance is an essential avenue for exploration.

4. \*\*Advanced Optimization Algorithms\*\*:

- The choice of optimization algorithm can significantly impact convergence speed and final results. While we used the Adam optimizer, exploring alternative algorithms like SGD with momentum or adaptive learning rate schedules could potentially lead to more robust convergence and lower loss values.

5. \*\*Data Augmentation Refinement\*\*:

- Further refining data augmentation strategies is essential. Experimenting with different augmentation techniques, such as Gaussian noise injection or color jittering, can introduce additional diversity into the training dataset, potentially improving the model's ability to generalize.

6. \*\*Transfer Learning\*\*:

- Transfer learning from pre-trained models on larger datasets like ImageNet can be a powerful approach. Fine-tuning pre-trained models on CIFAR-10 can provide a significant boost in accuracy and convergence speed. This method is particularly effective when labeled data is limited.

7. \*\*Ensemble Methods\*\*:

- Employing ensemble methods, such as stacking or bagging, can help amalgamate the predictions of multiple models to achieve a higher overall accuracy. Combining the outputs of several well-optimized models can lead to robust and reliable results.

8. \*\*Class Imbalance Handling\*\*:

- Analyzing class imbalances within the CIFAR-10 dataset is crucial. Certain classes may have more examples than others, leading to biased predictions. Addressing class imbalances through techniques like oversampling or class weighting can help improve overall model performance.

9. \*\*Advanced Loss Functions\*\*:

- Exploring advanced loss functions tailored to the specific characteristics of the CIFAR-10 dataset can be beneficial. Loss functions like focal loss or class-weighted loss can be more effective in handling imbalanced datasets and challenging classes.

10. \*\*Interpretable AI\*\*:

- Developing methods for interpreting model predictions is increasingly important, especially in applications where decisions impact human lives. Future research can focus on creating tools and techniques for explaining the rationale behind the model's classifications, enhancing its transparency and trustworthiness.

11. \*\*Computational Efficiency\*\*:

- As deep learning models become more complex, ensuring computational efficiency becomes paramount. Future work can involve optimizing model inference and training pipelines for deployment in resource-constrained environments.

12. \*\*Alternative Datasets and Domains\*\*:

- Expanding the research scope to encompass alternative datasets and domains can lead to a more comprehensive understanding of neural network performance. Investigating different image datasets or even exploring applications in natural language processing or reinforcement learning could broaden the research's impact.

\*\*Conclusion\*\*:

In conclusion, our research journey into training a neural network on the CIFAR-10 dataset has yielded promising results, but it also illuminates the vast opportunities for further advancements. The achieved accuracy of approximately 53% demonstrates the potential of our model, especially given the dataset's intricacies. However, we acknowledge that this is merely the beginning of our pursuit for excellence in image classification.

As we navigate the labyrinth of possibilities in future work, we emphasize the significance of exploring more intricate model architectures, fine-tuning hyperparameters, embracing regularization techniques, and optimizing optimization algorithms. Data augmentation, transfer learning, and ensemble methods also stand as formidable avenues for elevating our model's performance.

Moreover, we advocate for a holistic approach to research that goes beyond accuracy metrics. The interpretability of AI models, fairness, and ethical considerations are becoming increasingly important. Ensuring that our models make decisions that are transparent and equitable is a crucial aspect of AI research.

Ultimately, our research is a stepping stone towards building more intelligent and capable AI systems. By diligently pursuing these future directions, we endeavor to contribute not only to the field of computer vision but also to the broader landscape of artificial intelligence, where the potential for transformative impact is vast and exciting.

# References

Lim, S.-H., Young, S.R. and Patton, R.M. (2016). An analysis of image storage systems for scalable training of deep neural networks.

Majumder, U., Christiansen, E., Wu, Q., Inkawhich, N., Blasch, E. and Nehrbass, J. (2017). High-performance computing for automatic target recognition in synthetic aperture radar imagery. *Cyber Sensing 2017*. doi:https://doi.org/10.1117/12.2263218.

Das, K., Conjeti, S., Chatterjee, J. and Sheet, D. (2020). Detection of Breast Cancer From Whole Slide Histopathological Images Using Deep Multiple Instance CNN. *IEEE Access*, 8, pp.213502–213511.

Zhu, Z., Tan, L., Li, Y. and Ji, C. (2020). PHDFS: Optimizing I/O performance of HDFS in deep learning cloud computing platform. *Journal of Systems Architecture*, [online] 109, p.101810.

Lu, S., Wei, X., Li, Y. and Wang, L. (2018). Detecting Anomaly in Big Data System Logs Using Convolutional Neural Network. *Dependable Autonomic and Secure Computing*.

Zhang, H., Huang, H. and Wang, L. (2017). MRapid: An Efficient Short Job Optimizer on Hadoop. *International Parallel and Distributed Processing Symposium*.

Dataset reference:

Krizhevsky, A. (2009). *Learning Multiple Layers of Features from Tiny Images*

Learning outcomes that they’re testing for:

Big data:

* Critically assess the data storage and management requirements of a given data project from a modern perspective and evaluate limitations of legacy approaches to Big Data
  + **Talk about the data I have, and how I’m going to store and process it and why old storage techniques are not useful.**
* Assess the design concepts and architectural patterns of distributed Big Data systems and analyse the components that form their technology stack
  + **Check a few distributed Big Data Systems (Hadoop/ NoSQL Databases/Stream processing/Data Warehousing/Container Orchestration**
* Critically evaluate and select a Big data environment suitable for retrieving and processing a given Big Data set, perform data management and select appropriate analytic algorithms for the required scale and speed.
  + **This is the actual doing of the experiment. Talk about why Hadoop was chosen for this particular project**

Advanced Data Analytics:

* Debate the theory and application of different types of neural networks
  + **Talk about different Neural network types and where they’re useful**
* Analyse a set of requirements to determine the type of Neural Network for a particular problem set. Document and justify the choices made to stakeholders and peers through insight gained from the process.
  + **Talk about why I chose the Neural Network I chose**

Certainly! Here's an example of a data analytics project that incorporates Hadoop and image classification using Convolutional Neural Networks (CNNs):

\*\*Project Title: Big Data Image Classification with Distributed CNNs using Hadoop\*\*

\*\*Project Overview:\*\*

In this project, you will develop an image classification system using CNNs and leverage Hadoop for distributed data processing. The objective is to classify a massive dataset of images into predefined categories while efficiently handling the scale of big data.

\*\*Project Steps:\*\*

1. \*\*Data Collection and Storage:\*\*

- Gather a large-scale image dataset, potentially containing millions of images. This dataset should be stored on Hadoop Distributed File System (HDFS) for efficient distributed processing.

2. \*\*Data Preprocessing:\*\*

- Preprocess the image data by resizing, normalizing, and augmenting as necessary.

- Convert the images into a format suitable for Hadoop processing, like Hadoop SequenceFile or Parquet.

3. \*\*Hadoop Distributed Processing:\*\*

- Utilize Hadoop MapReduce or Spark to distribute the preprocessing and feature extraction tasks across a Hadoop cluster.

- You can use deep learning libraries like TensorFlow or PyTorch on each node to apply pre-trained CNN models for feature extraction.

4. \*\*Neural Network Model for Image Classification:\*\*

- Design and train a CNN model for image classification using a deep learning framework.

- Transfer learning can be particularly useful in this scenario, where you fine-tune a pre-trained model on the extracted features from Hadoop.

5. \*\*Model Training and Validation:\*\*

- Train your CNN model on a subset of the data to validate its performance.

- Utilize cross-validation techniques to ensure robustness.

6. \*\*Distributed Model Training with Hadoop:\*\*

- Distribute the model training across your Hadoop cluster. This can be achieved using Hadoop's distributed computing capabilities.

7. \*\*Model Evaluation:\*\*

- Evaluate the trained model's performance using metrics such as accuracy, precision, recall, and F1-score on a separate validation dataset.

8. \*\*Inference on Big Data:\*\*

- Deploy the trained model to perform inference on the large-scale image dataset stored in HDFS.

- Utilize Hadoop to parallelize and distribute inference tasks across the cluster, allowing for efficient processing.

9. \*\*Scaling for Real-Time Inference:\*\*

- If real-time inference is required, integrate the model with a stream processing framework (e.g., Apache Kafka and Apache Flink) for handling incoming image data in real-time.

10. \*\*Monitoring and Maintenance:\*\*

- Implement monitoring and maintenance routines to ensure the system's reliability and performance in a production environment.

\*\*Skills and Technologies Involved:\*\*

- Convolutional Neural Networks (CNNs)

- Deep learning frameworks (e.g., TensorFlow, PyTorch)

- Hadoop ecosystem (HDFS, MapReduce, Spark)

- Data preprocessing and augmentation techniques

- Model evaluation and fine-tuning

- Distributed computing and cluster management

- Real-time stream processing (if applicable)

This project demonstrates the integration of Hadoop's distributed processing capabilities with CNNs for scalable image classification on big data. It addresses the challenges of processing and analyzing large-scale image datasets efficiently.