*A project report on*

# **CYCLONE INTENSITY PREDICTION USING DEEP LEARNING**

# **ABSTRACT**

This research paper explores the application of deep learning techniques in the field of cyclone intensity prediction, specifically utilizing five distinct models: Alexnet, VGG16, Basic CNN, Deepphurie and a novel hybrid model that combines image and numerical data. The first part of our study investigates the performance of the Alexnet, VGG16 and Deepphurie convolutional neural networks (CNNs) in processing satellite imagery data. These models are trained on a comprehensive dataset of cyclone images, enabling them to learn complex features and patterns associated with cyclone formation and intensity. The results indicate that these deep learning models can effectively extract relevant features from the images, contributing to the accurate prediction of cyclone intensity. In addition to image data, we incorporate numerical meteorological data into the prediction process. To this end, we introduce a hybrid model that combines the strengths of both image-based and numerical data-driven predictions. This model leverages the capabilities of recurrent neural networks (RNNs) to process time-series meteorological information in conjunction with CNNs for image analysis. This research provides a substantial contribution to the field of meteorology and disaster management, offering a foundation for the development of more robust and accurate cyclone intensity forecasting systems, ultimately aiding in the protection of vulnerable coastal communities.

# **1. INTRODUCTION**

Cyclones, also known regionally as hurricanes or typhoons, unleash a devastating cocktail of ferocious winds, torrential rains, and storm surges, leaving coastal communities in their wake to grapple with immense destruction. The increasing frequency and intensity of these storms due to climate change necessitate a significant leap forward in our ability to predict their exact strength. Traditional image based weather models like those in [1]-[15] and numerical based weather models like those in [22]-[30], while adept at forecasting cyclone tracks, often fall short when it comes to precise intensity prediction. This impedes crucial preparedness efforts and disaster mitigation strategies, leaving communities vulnerable.

In recent years, advancements in machine learning, particularly the powerful subfield of deep learning, have revolutionized various scientific disciplines. Deep learning models like the ones used in [1], [4]-[10] and [14], with their ability to extract complex patterns and relationships from vast datasets, offer a promising new approach to cyclone intensity prediction. This research delves into the potential of deep learning to unlock previously inaccessible insights into the intricate dynamics of these tropical systems.

This study investigates the efficacy of a diverse set of deep learning models as seen in [1], [6], [10]- [21] for improved cyclone intensity prediction. We explore the performance of established architectures like Basic CNN[8], VGG16 [14] and AlexNet [19], which have proven successful in other image recognition tasks. Additionally, we introduce a DeepPhurie model specifically designed for this application. Furthermore, to leverage the potential benefits of incorporating multi-modal information, a hybrid model that utilizes both satellite imagery and numerical data is evaluated. By comprehensively assessing and comparing the performance of these models, we aim to identify the most effective deep learning approach for cyclone intensity prediction.

The objectives of this research are multifaceted. First and foremost, we seek to evaluate the performance of deep learning models in the context of cyclone intensity prediction. This involves assessing their capacity to learn and exploit relevant features from the diverse data sources available, including satellite imagery, radar data, and historical storm records. We aim to quantify the accuracy of these models and investigate their potential to outperform or complement traditional numerical modeling techniques. Additionally, we delve into the concept of transfer learning, exploring how pre-trained models can be fine-tuned for meteorological applications.

Beyond the standalone evaluation of CNN models, we explore the synergistic potential of the hybrid model that combines both image and numerical data. This approach capitalizes on the idea that the fusion of complementary information sources can lead to more robust and accurate predictions. By integrating visual data that captures the evolving structure of cyclones with numerical data that encodes their environmental context, we anticipate improvements in prediction accuracy. This novel approach has the potential to enhance the forecasting of cyclone intensity and provide valuable insights into the intricate processes governing these extreme weather events. Through this comprehensive evaluation and exploration, this research contributes to the ongoing efforts to advance cyclone intensity prediction. The findings presented here have the potential to enhance early warning systems, inform disaster management strategies, and ultimately mitigate the devastating impacts of cyclones on vulnerable communities and ecosystems.

The structure of this paper proceeds as follows: Chapter 2 provides a thorough review of the existing literatures, highlighting the state of the art in cyclone intensity prediction and the role of deep learning in meteorology. In Chapter 3, we elaborate on the methodology employed, including data collection, pre-processing, and the architecture of the deep learning models under investigation. Chapter 4 presents the results of our experiments, supported by quantitative metrics and visualizations, shedding light on the performance of AlexNet, VGG16, CNN, Deep-Phurie and the Hybrid model.

In Chapter 5 and 6, we draw conclusions and offer insights into the potential for future research in the field of cyclone intensity prediction using deep learning. Ultimately, this research aspires to contribute towards saving lives and minimizing property damage by enabling communities to better anticipate and respond to the destructive forces of cyclones.

# **3. METHODOLOGY**

The project was divided into various steps in order to have a efficient and well organized workflow. All the steps and related details have been listed below:

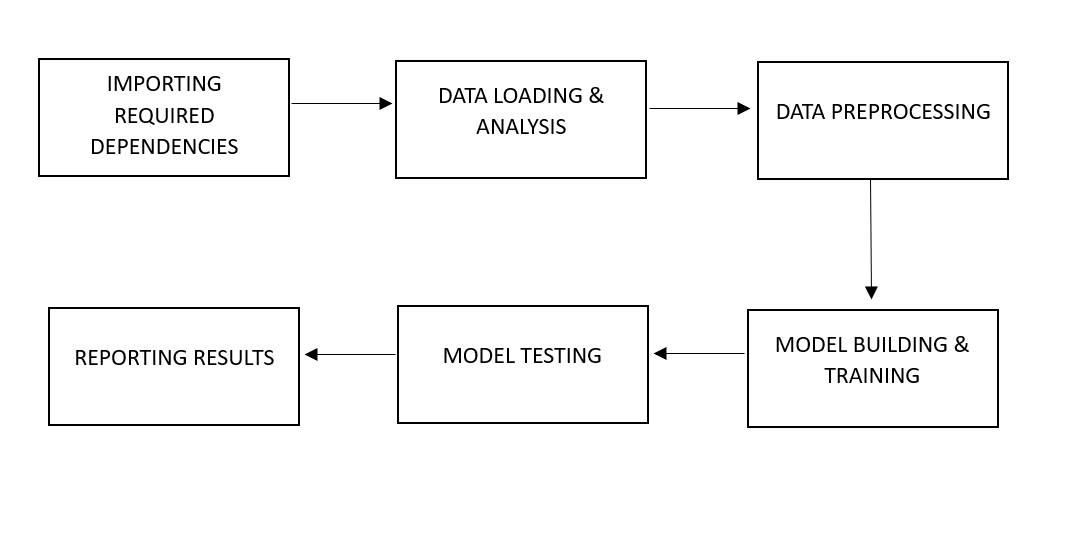


Figure 1: Work Flow Diagram for the project

## **3.1 IMPORTING REQUIRED LIBRARIES**

Various libraries were used for the data manipulation, data processing, model building etc. But the main ones were numpy, pandas, tensorflow, sklearn and keras. NumPy, short for “Numerical Python,” which was used in [2], [4] and [5], is a fundamental library for numerical and scientific computing in the Python programming language. It was created by Travis Oliphant in 2005 and has since become an essential tool for data scientists, engineers, and researchers in various fields. NumPy provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays efficiently. At the core of NumPy are its nd array objects, which are N-dimensional arrays that can store elements of the same data type. These arrays are much more efficient than Python’s built-in lists for numerical computations because they are implemented in C and allow for vectorized operations. Vectorized operations mean that you can apply operations to entire arrays or even perform array-to-array operations without the

need for explicit Python for loops. This capability leads to faster and more concise code, making NumPy a powerful tool for tasks such as data manipulation, statistical analysis, and numerical simulations. NumPy also offers a vast library of mathematical functions, including those for linear algebra, Fourier analysis, random number generation, and more. Moreover, it provides tools for integrating C/C++ and Fortran code, enhancing its versatility for performance-critical applications. With its open-source nature and an active community, NumPy continues to evolve, ensuring that Python remains a top choice for numerical and scientific computing. It also serves as the foundation for many other scientific computing libraries, such as SciPy, scikit-learn, and pandas, making it an integral part of the Python data science ecosystem.

Pandas, used in [1], [21]-[27] and [31], is a popular and powerful open-source data manipulation and analysis library for the Python programming language. It was created by Wes McKinney and first released in 2008, making it an essential tool in the toolkit of data scientists, analysts, and engineers for handling and analyzing structured data. Pandas provides data structures and functions that simplify the process of data cleaning, transformation, and exploration, making it an indispensable tool in the field of data science. The two primary data structures in Pandas are the Data Frame and Series. A Data Frame is essentially a two-dimensional table that can store data in a tabular form, with rows and columns, while a Series is a one-dimensional array-like object. Pandas allows you to perform operations like indexing, slicing, filtering, and aggregating data effortlessly, making it particularly well-suited for data preparation and manipulation tasks. Pandas is also renowned for its compatibility with other data analysis and machine learning libraries, such as NumPy, Matplotlib, and scikit-learn. This seamless integration allows data scientists to create comprehensive data analysis pipelines with ease. Moreover, Pandas excels at handling missing data and provides a wide array of functions for merging, reshaping, and pivoting data, making it a versatile and efficient tool for various data-related tasks. Its straightforward and intuitive syntax makes it accessible to both beginners and experienced data professionals, contributing to its widespread adoption in the data science community.

TensorFlow, used in [5]- [10], [15] and [17], is an open-source machine learning framework developed by Google that has gained widespread popularity in the fields of artificial intelligence and deep learning. It was first released in 2015 and has since become one of the most widely used and respected tools for building and training machine learning models. TensorFlow is designed to be flexible and scalable, making it suitable for a wide range of

applications, from simple regression tasks to complex deep neural networks. One of the key strengths of TensorFlow is its ability to handle deep learning models with ease. It provides a high-level API called Keras, which simplifies the process of building and training deep neural networks. This high-level API allows researchers and developers to experiment with different network architectures and hyperparameters while abstracting away many of the low-level implementation details. This flexibility has made TensorFlow a popular choice for both beginners and experts in the field of deep learning. TensorFlow also offers strong support for deploying machine learning models in production environments. It has tools and libraries for serving models over the web, on mobile devices, and even on edge devices, making it a versatile framework for real-world applications. Additionally, TensorFlow supports hardware acceleration, enabling the use of GPUs and TPUs (Tensor Processing Units) to speed up training and inference, making it one of the most powerful and efficient frameworks available for deep learning tasks. Overall, TensorFlow’s combination of flexibility, scalability, and ease of use has made it a cornerstone in the machine learning and artificial intelligence community. Its extensive ecosystem of libraries and tools, along with a dedicated community of developers, continues to drive advancements in the field of deep learning and AI, making it a valuable resource for researchers, data scientists, and engineers working on machine learning projects.

Sklearn, often called as scikit-learn, used in [9], [17]- [26] and [30], is an open-source machine learning library for the Python programming language. It is one of the most popular and widely used tools for data science and machine learning, known for its simplicity, versatility, and extensive range of algorithms and tools. Developed by a large community of contributors, scikit-learn provides a user-friendly and consistent interface for a wide variety of machine learning tasks, including classification, regression, clustering, dimensionality reduction, and more. One of the key strengths of scikit-learn is its ease of use. It offers a straightforward and well-documented API that makes it accessible to both beginners and experienced machine learning practitioners. This simplicity allows users to quickly prototype and implement machine learning models without having to write complex code from scratch. Additionally, scikit-learn integrates seamlessly with other popular Python libraries like NumPy, pandas, and Matplotlib, making it a natural choice for data scientists and engineers working in Python-based environments. Scikit-learn includes a rich selection of machine learning algorithms, such as support vector machines, decision trees, random forests, k-nearest neighbors, and many more. Furthermore, it provides tools for data preprocessing, feature selection, and model evaluation, simplifying the entire machine learning pipeline. The library also supports hyperparameter tuning and cross-validation to help users optimize their models effectively. Moreover, it offers a comprehensive set of utilities for handling imbalanced data, which is a common challenge in real-world machine learning applications. In summary, scikit-learn is a powerful and user-friendly machine learning library that has played a pivotal role in democratizing machine learning and making it accessible to a broader audience. Its simplicity, extensive documentation, and rich set of algorithms have made it a go-to choice for data scientists and machine learning practitioners worldwide. Whether you are a beginner looking to get started with machine learning or an experienced professional working on complex projects, scikit-learn is an invaluable tool for developing and deploying machine learning solutions in Python.

Keras, used in [6]-[14], [19] and [29], is a popular and user-friendly open-source deep learning framework that has gained widespread adoption in the field of artificial intelligence and machine learning. Originally developed as an independent project by François Chollet, Keras has since been integrated into the TensorFlow library, making it one of the most accessible and versatile tools for building and training neural networks. One of Keras’ standout features is its high-level, user-friendly API that simplifies the process of creating and training deep learning models. It provides a clear and intuitive interface for defining neural network architectures, which allows both beginners and experienced researchers to rapidly prototype and experiment with various deep learning models. The simplicity of Keras makes it an ideal choice for teaching machine learning concepts, as it abstracts away many of the complexities involved in neural network development. Keras supports a wide range of neural network architectures, including feedforward networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more. Additionally, it offers pre-trained models and a wealth of layers, activation functions, and optimization algorithms to choose from, making it easy to tailor a model to specific tasks. Its seamless integration with TensorFlow also allows users to harness the power of TensorFlow’s low-level functionality while retaining the convenience of Keras’ high-level interface. Moreover, Keras has been instrumental in democratizing deep learning, enabling researchers, engineers, and data scientists to develop cutting-edge AI solutions with ease. Its active community and extensive documentation have contributed to its popularity and growth, making it an essential tool for anyone interested in the field of deep learning. Whether you are building neural networks for computer vision, natural language processing, or other machine learning tasks, Keras remains a valuable and approachable resource in the deep learning toolbox.

## **3.2 DATA LOADING & ANALYSIS**

The dataset we had was around 4 GB in size, containing various numerical values like latiitue, longitude, Eye Location and corresponding images. The whole dataset was in h5 format. The H5 file format, short for Hierarchical Data Format version 5, is a versatile and widely used data storage format in the scientific and data analysis communities. H5 files are designed to efficiently store and organize large datasets, making them ideal for applications in fields like astrophysics, bioinformatics, climate modeling, and machine learning. What sets H5 apart is its hierarchical structure, which allows for the storage of diverse data types, from numerical arrays to metadata and even user-defined attributes. This hierarchical organization makes it easy to manage, access, and share complex datasets, fostering collaboration and data sharing among researchers and analysts. Additionally, H5 files can be accessed using various programming languages, such as Python and MATLAB, thanks to libraries like h5py, simplifying data manipulation and analysis across different platforms. This format's adaptability and scalability have solidified its place as a go-to choice for storing and managing diverse datasets in the scientific and data science domains.

The dataset was gathered from sources like GridSat(images from geostationary satellites with various channels that were gathered since a long time), CMORPH(provided data from low orbit satellites transmitted through spatial propagation). It also contained information like our main predictor variable i.e intensity(maximum sustained wind speed, in knots), size of cyclone, minimum sea-level pressure, cyclone’s eye location.The images were present in 4 different channels. First was, infrared channel, also known as IR images, are a valuable tool in various fields, including remote sensing, meteorology, and thermal imaging according to [2], [3] and [7]. These images capture the invisible, long-wavelength infrared radiation emitted by objects and surfaces, providing unique insights into temperature variations and material properties. In remote sensing, IR imagery aids in environmental monitoring, crop health assessment, and disaster management by revealing temperature anomalies and differentiating between land and water. In meteorology, IR images help track cloud formations, measure surface temperatures, and identify weather patterns, enhancing weather forecasting capabilities. Additionally, thermal IR images are indispensable in industrial and scientific applications, offering a non-invasive means to detect heat anomalies, structural defects, and energy inefficiencies. Overall, infrared channel images play a crucial role in our ability to see and understand the world beyond the limits of human vision.

Second was, water vapor channel, these images are a valuable tool in meteorology and remote sensing, providing a unique perspective on atmospheric conditions. These images capture the distribution and movement of water vapor in the Earth's atmosphere, allowing meteorologists to monitor the development of weather systems, such as storms, fronts, and moisture patterns. By detecting variations in water vapor concentration, these images aid in the prediction of weather phenomena and provide essential information for forecasting. Water vapor channel images are particularly useful for tracking the evolution of severe weather events, as they reveal the intricate dynamics of moisture transport and atmospheric instability, helping scientists and forecasters better understand and anticipate changes in our ever-changing climate. Third was, visible channel, commonly referred to as optical images or simply "what the eye sees," capture the world as it appears to the human eye. These images rely on the portion of the electromagnetic spectrum that includes visible light, spanning a range of wavelengths from approximately 400 to 700 nanometers. The information conveyed by visible channel images is fundamental for numerous applications, from everyday photography and cinematography to meteorology, remote sensing, and astronomy. These images are rich in color and detail, making them a valuable tool for capturing and conveying visual information in a format that is easily comprehensible to humans. They offer a window into the world as we perceive it, revealing a vibrant tapestry of colors, shapes, and textures that shape our understanding of the natural and man-made environments around us. These were unstable because of the daylight.Fourth was, Passive micro wave channel, Passive microwave channel images are a valuable tool in remote sensing and Earth observation, offering a unique perspective on our planet's surface and atmospheric conditions. These images are generated by measuring the naturally emitted microwave radiation from the Earth's surface and its atmosphere, providing critical data for a wide range of applications. Passive microwave sensors can peer through clouds, rain, and even darkness, making them indispensable for monitoring weather patterns, tracking sea ice extent, studying soil moisture, and understanding the Earth's energy budget. By capturing the microwave emissions at various frequencies, researchers can extract essential information about temperature, humidity, and other environmental parameters. This non-invasive and all-weather capability of passive microwave channel images contributes significantly to our understanding of the Earth's dynamic processes and aids in making informed decisions for environmental and climate management. Other than that images were of 201\*201 size, with radius of 7 degrees in both latitude and longitude. It’s center was places in the middle of the vector, with distance of about 4km between two data points. It had a resolution of 7/100 degree lat/lon.

## **3.3 DATA PREPROCESSING**

Data preprocessing is a critical and fundamental step in the field of data science and machine learning. It involves the cleaning, transformation, and organization of raw data to make it suitable for analysis and modeling. This process includes tasks such as handling missing values, scaling or normalizing features, encoding categorical variables, and removing outliers. Data preprocessing not only ensures the quality and reliability of the dataset but also contributes to the overall success of predictive models and analytics by reducing noise and enhancing the meaningful information that can be extracted from the data. Effective data preprocessing paves the way for more accurate and robust results, making it an indispensable stage in any data-driven project.

There were a lot of preprocessing techniques used in the reference articles that we used in the research. While [1]-[15] and [21] dealt with image data preprocessing techniques like cropping and rotating of satellite images, [22]-[30] dealt with preprocessing of numerical data like removing null values, removing irrelevant columns like time of cyclone, date of cyclone and cyclone IDs in order to enhance the processing power of the models.

For our paper we only did basic steps in order to avoid too much complexity for the model. First we removed all NaN containg rows, then from the remaining data only IR and PMW channel images were taken, as they were stable. Now this data was divided into train and test data in 80:20 proportion, respectively. The images were also converted into tensors and numerical data was standardized for easier model training. After this image rotation and center cropping was done, in order to get better trained model, and get accurate results even in case of abnormalities.

## **3.4 MODEL BUILDING & TRAINING**

In our paper we have proposed 2 novel models, and other than that we also built traditional/pretrained models like alexnet & vgg16, and compared our models with them. The model details, architechture and data flow diagrams are shown further into the paper. Also, for all the models a Adam optimizer was used. A model optimizer is a critical component in the field of machine learning and deep learning. Its primary function is to enhance the efficiency and performance of neural network models. Model optimizers work by applying various mathematical and computational techniques to streamline the model's architecture, reducing its size and computational complexity while preserving or even improving its predictive capabilities. These optimizations help to make models more practical for deployment in resource-constrained environments, such as mobile devices and edge devices, where computational resources are limited. By striking a balance between model accuracy and computational efficiency, model optimizers play a pivotal role in accelerating the adoption of AI and machine learning across a wide range of applications, from image recognition to natural language processing.

The Adam optimizer, used in [3]-[9] and [13], short for Adaptive Moment Estimation, is a popular optimization algorithm used in training machine learning models, particularly deep neural networks. It was introduced by Diederik P. Kingma and Jimmy Ba in their 2014 paper titled "Adam: A Method for Stochastic Optimization." Adam is a versatile and powerful optimization technique that combines the advantages of two other commonly used optimization methods: stochastic gradient descent (SGD) and RMSprop (Root Mean Square Propagation). Adam is known for its efficiency and effectiveness in optimizing a wide range of machine learning models. It maintains two moving averages for each parameter being optimized: the first moment (mean) and the second moment (uncentered variance). These moving averages help adapt the learning rate for each parameter individually, making the algorithm suitable for

non-stationary and noisy environments. The key idea behind Adam is to dynamically adjust the learning rates for different parameters based on their historical gradients. The algorithm employs a combination of gradient information and adaptive learning rates to converge faster and reach better solutions. It computes exponentially weighted averages of past gradients and their squares, which are then used to adjust the learning rate for each parameter. This adaptability makes Adam well-suited for deep learning tasks, where the landscape of the optimization problem can be complex and non-uniform. While Adam has proven to be highly effective in many applications, it's essential to fine-tune its hyperparameters, such as the learning rate, to achieve the best performance for a specific task. Overall, Adam has become a popular choice in the deep learning community, and its widespread adoption has contributed to the success of various state-of-the-art machine learning models. Now we will be discussing about the models that we have used in our research:

**3.4.1 ALEXNET**

AlexNet, as seen in [19], is a pioneering deep convolutional neural network (CNN) architecture that played a crucial role in the resurgence of artificial neural networks and their applications, particularly in the field of computer vision. Developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, this architecture made a significant impact when it won the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a prestigious competition in image classification and object detection. AlexNet was revolutionary for several reasons. First, it introduced a deep neural network with multiple convolutional and fully connected layers, which allowed it to capture intricate features and patterns in images. Second, it made use of the rectified linear unit (ReLU) activation function, which helped accelerate training and mitigate the vanishing gradient problem. Additionally, AlexNet utilized techniques like dropout to reduce overfitting and data augmentation to enhance the model's robustness. This network architecture consists of eight layers, including five convolutional layers and three fully connected layers. It used a large number of parameters, which required significant computational power for training. Its success showcased the potential of deep learning in computer vision tasks, spurring further research and development in the field. AlexNet's innovations have since paved the way for more advanced CNN architectures, such as VGG, GoogLeNet, and ResNet, contributing to the rapid progress in image recognition, object detection, and various other visual recognition applications.

AlexNet consists of eight layers: five convolutional layers followed by three fully connected layers. The convolutional layers are responsible for learning features at different levels of abstraction, and the fully connected layers are used for classification. It employs a ReLU (Rectified Linear Unit) activation function, which helps accelerate convergence during training. AlexNet was also one of the first deep networks to implement dropout as a regularization technique to prevent overfitting, where randomly selected neurons are dropped out during training to reduce co-adaptation of features.

Rectified Linear Unit (ReLU), used in [4], [5], [8] and [10], is a widely used activation function in artificial neural networks, particularly in deep learning models. It is a simple yet effective non-linear function that introduces the concept of thresholding in neural network neurons. When the input to a ReLU neuron is positive, it allows the signal to pass through unchanged, essentially acting as a linear function. However, when the input is negative, it simply outputs zero, effectively turning off the neuron. This piecewise linearity and sparsity-inducing property of ReLU helps mitigate the vanishing gradient problem and accelerates the

training of deep neural networks. ReLU has become a cornerstone of modern deep learning architectures, contributing to their ability to learn complex and hierarchical representations from data.

Max pooling is a fundamental technique in the field of convolutional neural networks (CNNs) used for image and data processing tasks. It plays a crucial role in down-sampling the spatial dimensions of feature maps, reducing computational complexity, and extracting the most important information from an input image. In max pooling, a small window or filter slides over the input data, typically in a 2x2 or 3x3 grid, and selects the maximum value from each group of adjacent values. This process results in a reduced-resolution feature map with preserved dominant features, aiding in translation and rotation invariance while improving computational efficiency. Max pooling is a valuable component of deep learning architectures, enhancing the network’s ability to detect and represent hierarchical features within images, making it a key technique for image classification and object detection tasks.

While training this model we also used kfold training method. K-fold cross-validation is a widely used technique in machine learning for assessing the performance and generalization capabilities of a model. It involves splitting a dataset into K subsets of approximately equal size, typically K being set to 5 or 10. The model is trained and evaluated K times, with each fold serving as a validation set while the remaining K-1 folds are used for training. This process ensures that every data point is used for both training and validation at least once.

The results from each fold are then averaged to provide a more robust estimate of the model’s performance, reducing the risk of overfitting and helping to gauge how well the model might perform on unseen data. K-fold cross-validation is a valuable tool for model selection and hyperparameter tuning, enhancing the reliability of machine learning algorithms.

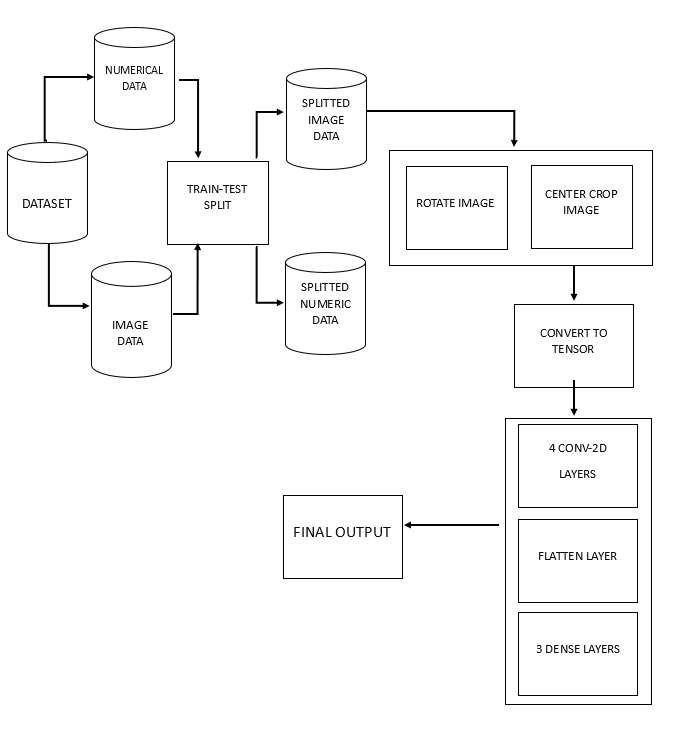


Figure 2: Data Flow Diagram for AlexNet

**3.4.2 VGG16**

VGG16, used in [14], short for “Visual Geometry Group 16,” is a convolutional neural network architecture that has had a significant impact on the field of computer vision and deep learning. It was developed by the Visual Geometry Group at the University of Oxford and was one of the top-performing models in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a prestigious competition in the field of computer vision. The architecture of VGG16 is characterized by its simplicity and effectiveness. It consists of 16 weight layers, including 13 convolutional layers and 3 fully connected layers. VGG16 is known for its use of small 3x3 convolutional filters with a stride of 1 and zero-padding to maintain spatial dimensions. The network’s architecture follows a pattern of stacking multiple convolutional layers with max-pooling layers in between, which helps in progressively reducing the spatial dimensions of the input while increasing the depth of feature extraction. The final fully connected layers serve as a classifier for various object categories. One of the key strengths of VGG16 is its exceptional performance in image classification and object recognition tasks. Its deep architecture allows it to capture a wide range of image features, making it adept at distinguishing fine-grained details in images. However, this depth also makes VGG16 computationally intensive and less efficient in terms of memory and speed compared to more modern architectures like ResNet and Inception. Researchers have built upon the ideas from VGG16 to develop more streamlined and efficient deep neural networks while maintaining high accuracy in various computer vision tasks. Despite its age, VGG16 remains a foundational model in the deep learning community, and its architecture has inspired many subsequent designs. It showcases the importance of depth in convolutional neural networks for feature extraction, and its legacy lives on as a benchmark for image classification and understanding in the field of computer vision. Softmax activation, used in [14], is a widely used mathematical function in the field of deep learning and neural networks. It serves as a key component for transforming a vector of raw scores or logits into a probability distribution. The softmax function takes an input vector and exponentiates each element while normalizing the result to ensure that the output values sum up to one. This normalization process makes softmax particularly useful for multi-class classification problems, as it allows the model to assign a probability to each class, indicating the likelihood of the input belonging to that class. Softmax is commonly employed in the output layer of neural networks for tasks such as image classification, natural language processing, and more, making it a fundamental tool for converting model predictions into interpretable and actionable class probabilities. The Flatten layer is a crucial component in neural networks, particularly in deep learning and convolutional neural networks (CNNs). Its primary purpose is to transform the multi-dimensional output of the preceding layers, typically 2D or 3D arrays like feature maps, into a one-dimensional vector. This transformation is essential when transitioning from convolutional and pooling layers to fully connected layers in the network. By flattening the data, the Flatten layer ensures that the subsequent layers can process it as a flat input, enabling tasks like classification and regression. In essence, it simplifies the complex spatial information captured by earlier layers into a format suitable for traditional neural network operations, making it a fundamental element in building effective and efficient deep learning model.

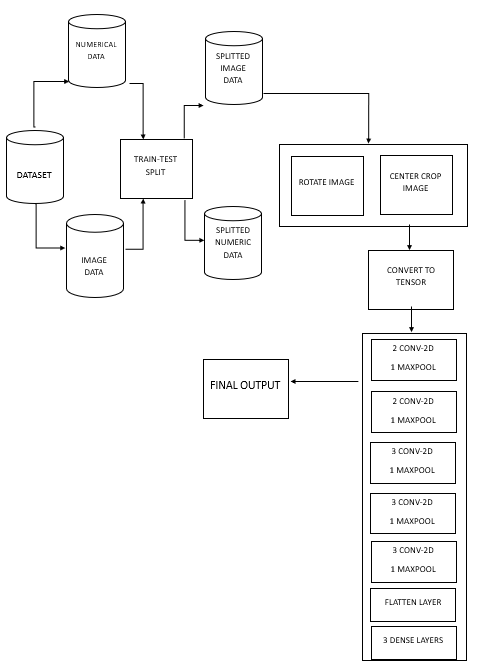


Figure 3: Data Flow Diagram for VGG16

**3.4.3 BASIC CNN**

This study incorporates a basic convolutional neural network (CNN) architecture, which was seem in [8] specifically tailored for cyclone intensity prediction. CNNs have demonstrated remarkable success in various image recognition tasks, and this model leverages their ability to extract hierarchical features fromsatellite imagery of cyclones.

The proposed CNN architecture comprises of a sequence of convolutional layers followed by fully-connected layers. The convolutional layers act as feature extractors, progressively learning increasingly complex and abstract representations of the cyclone’s visual characteristics within the satellite imagery.

**Convolutional Layers (4):**

* These layers employ filters or kernels that slide across the image, detecting and extracting local features like edges, textures, and patterns.
* Each convolutional layer utilizes multiple filters, generating a collection of feature maps highlighting different aspects of the cyclone’s visual information.
* The ReLU (Rectified Linear Unit) activation function is applied after each convolution, introducing non-linearity and accelerating training convergence.

**Flatten Layer (1):**

* Once the convolutional layers have processed the image, the extracted features are flattened into a single-dimensional vector. This vector serves as the input to the subsequent fully-connected layers.

**Dense Layers (5):**

* These fully-connected layers act as a classifier, tasked with transforming the flattened feature

vector into a prediction of the cyclone’s intensity.

* Each neuron in a dense layer is connected to all neurons in the preceding layer, allowing for

complex feature combination and classification.

* The ReLU activation function is employed throughout the dense layers to introduce non-

linearity and improve model capacity.

This basic CNN architecture offers a balance between model complexity and effectiveness.

It leverages the strengths of CNNs in feature extraction while maintaining a manageable number of parameters, making it suitable for training with cyclone image datasets. By learning informative representations from satellite imagery, this model aims to contribute to improved cyclone intensity prediction.

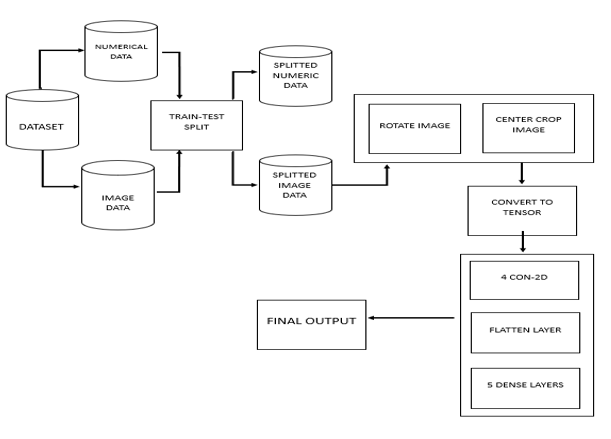


Figure 4: Data Flow Diagram for CNN

**3.4.4 DEEPPHURIE**

The DeepPhurie model, one of our novel models, is a convolutional neural network (CNN) architecture designed for cyclone intensity prediction. It leverages the power of CNNs in extracting features from satellite imagery of cyclones to predict their intensity.

**Network Architecture:**

* The model follows a sequential structure, stacking convolutional layers for feature extraction and fully-connected layers for classification.

**Convolutional Layers (6):**

* Six convolutional layers are employed, each using a kernel size of (3, 3) or (5, 5) to scan the input image and detect local features.
* The number of filters used in each layer is consistently set to 32, allowing the model to learn a diverse set of features from the cyclone imagery.
* After each convolution, a ReLU (Rectified Linear Unit) activation function is applied to introduce non-linearity and improve model capacity.

**Pooling Layers (6):**

* Following each convolutional layer, a max pooling layer with a pool size of (2, 2) or (3, 3) is used for downsampling.
* This process reduces the spatial dimensions of the feature maps while retaining the most important information, promoting computational efficiency and improving model generalization.

**Batch Normalization Layers (6):**

* Batch normalization layers are strategically placed after each convolutional layer.
* These layers help to normalize the activations across different mini-batches during training, accelerating the training process and improving model stability.

**Flatten Layer (1):**

* After the final convolutional layer, a flatten layer transforms the extracted features from a two-dimensional tensor into a single-dimensional vector.
* This vector serves as the input to the subsequent fully-connected layers.

**Dense Layers (2):**

* Two fully-connected layers with 256 and 128 neurons, respectively, are employed for classification.
* These layers aim to learn complex relationships between the extracted features and the cyclone’s intensity.
* Dropout with a rate of 0.5 is implemented after each dense layer to prevent overfitting by randomly dropping out neurons during training.
* ReLU activation functions are used throughout the dense layers to introduce non-linearity.

**Output Layer (1):**

* The final layer consists of a single neuron with a linear activation function.
* This neuron’s output represents the predicted intensity of the cyclone, likely a continuous value.

**Compilation:**

* The Adam optimizer is chosen for training the model due to its efficiency in handling various learning rate scenarios.
* Mean squared error (MSE) is used as the loss function, aiming to minimize the squared difference between the predicted and actual cyclone intensity values.

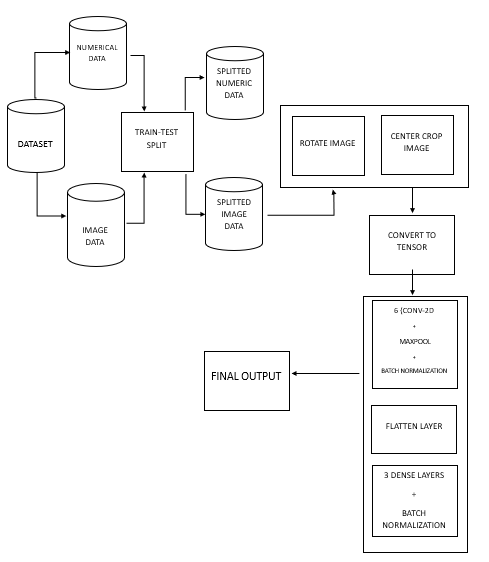


Figure 5: Data Flow Diagram for Deepphurie

Overall, the DeepPhurie model utilizes a deep convolutional architecture with batch normalization to effectively extract features from cyclone imagery. The subsequent dense layers with dropout regularization aim to learn the mapping between these features and the cyclone’s intensity. This model serves as one of the approaches investigated in this study for cyclone intensity prediction using deep learning.

**3.4.5 HYBRID MODEL**

This was also one of our novel models, which uses both image and numerical data for training and giving predictions. It uses CNN for image training and a sequential dense layer network for numerical data training. A Convolutional Neural Network, or CNN, is a specialized type of artificial neural network designed primarily for tasks involving visual data, such as image recognition and analysis. CNNs have revolutionized the field of computer vision and have found applications in a wide range of industries, from self-driving cars and medical imaging to facial recognition and object detection. CNNs are inspired by the structure and function of the human visual system. They consist of multiple layers of interconnected neurons, with each layer responsible for extracting different features from the input data. The core architectural elements of a CNN include convolutional layers, pooling layers, and fully connected layers. Convolutional layers are the heart of CNNs. They use a set of learnable filters or kernels to slide over the input data, performing convolution operations to extract features like edges, textures, and shapes.

Pooling layers, often in the form of max-pooling or average-pooling, help reduce the spatial dimensions of the data and emphasize the most important features. Fully connected layers, at the end of the network, combine these features to make final predictions. One of the key advantages of CNNs is their ability to automatically learn and hierarchically represent features from the raw data, reducing the need for handcrafted feature engineering. This makes them exceptionally effective in tasks like image classification, where they can identify and classify objects in images with remarkable accuracy. CNNs have also been extended to address more complex tasks, including object detection, image segmentation, and even generating new images through techniques like generative adversarial networks (GANs).

With ongoing research and advancements in deep learning, CNNs continue to play a pivotal role in advancing computer vision and are an essential component of many state-of-the-art machine learning applications.

A dense layer, often referred to as a fully connected layer, is a fundamental component in artificial neural networks, particularly in deep learning models like feedforward neural networks and multilayer perceptrons. In a dense layer, each neuron or node is connected to every neuron in the previous layer, creating a dense network of interconnected nodes. These connections have associated weights, which are adjusted during training to learn the relationships within the data. The output of a dense layer is typically computed through a linear combination of its inputs, followed by an activation function that introduces

non-linearity into the network. Dense layers play a crucial role in capturing complex patterns and features in data, making them an essential building block for a wide range of machine learning tasks, from image and speech recognition to natural language processing.

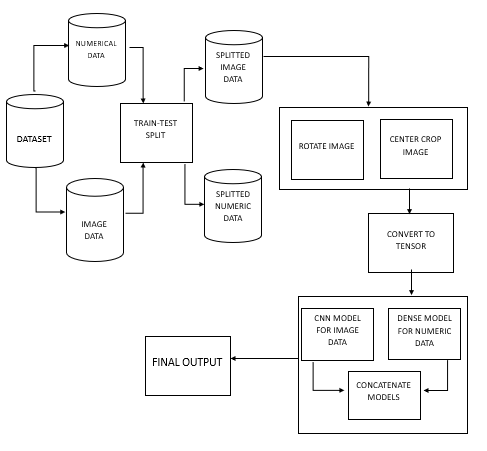


Figure 6: Data Flow Diagram for Hybrid Model

## **3.5 MODEL TESTING**

Model testing is a critical phase in the development and evaluation of various computational models, ranging from machine learning algorithms to complex simulations. During this stage, models are subjected to a series of carefully designed experiments or validation processes to assess their performance, accuracy, and generalizability. This process helps identify any shortcomings, errors, or biases within the model and allows researchers and engineers to

fine-tune and optimize it for real-world applications. Model testing is instrumental in ensuring that models can make reliable predictions, classifications, or decisions, thus increasing their usability and trustworthiness in various domains, such as healthcare, finance, and autonomous systems. It plays a vital role in validating the effectiveness and robustness of models, ultimately contributing to their successful deployment and impact.

We have used Mean Squared Error(MSE), which was used in [1]-[5] and [21], as our performance metrics. Mean Squared Error (MSE) and Mean Absolute Error (MAE), used in [6]-[11], are both metrics used to evaluate the performance of regression models, including those used in cyclone intensity prediction with deep learning. The choice between MSE and MAE depends on the specific characteristics of the problem and the priorities of the prediction task. Here's why MSE might be considered better than MAE for cyclone intensity prediction using deep learning:

**Emphasis on Larger Errors:** MSE penalizes larger errors more heavily compared to MAE. In the context of cyclone intensity prediction, accurately predicting extreme or high-intensity cyclones is often more critical than smaller errors in predicting lower-intensity cyclones. MSE’s emphasis on larger errors aligns well with this priority.

**Differentiability:** MSE is more conducive to optimization techniques like gradient descent due to its differentiability. Deep learning models, such as neural networks, often rely on gradient-based optimization algorithms for training. MSE provides smooth gradients throughout the optimization process, making it easier to find the minimum of the loss function.

**Mathematical Properties:** MSE is often preferred in statistical modeling and machine learning because it has desirable mathematical properties. For example, it arises naturally from maximum likelihood estimation under the assumption of Gaussian noise, which is a common assumption in regression tasks.

**Sensitivity to Outliers:** MAE gives equal weight to all errors, whereas MSE gives more weight to larger errors. If there are outliers in the data, MSE may be more robust as it downplays the effect of outliers compared to MAE.

# **4. RESULTS & DISCUSSION**

In order to compare our models, MSE(mean squared error) was calculated during both training and testing. Mean Squared Error (MSE) is a fundamental and widely used metric in statistics and machine learning for evaluating the accuracy of a predictive model or estimator. It provides a quantitative measure of how well a model's predictions match the actual observed data. MSE is particularly useful when dealing with regression problems, where the goal is to predict a continuous numeric value, such as stock prices, temperatures, or test scores. To compute MSE, you start by taking the squared difference between each predicted value and its corresponding actual value (often referred to as the "residuals"). These squared differences are then averaged across all data points. The formula for calculating MSE is:

**MSE = (1/n) \* Σ(predicted value - actual value)^2** (Equation 1)

Here, 'n' represents the number of data points in the dataset, and the Σ symbol denotes summation over all data points. MSE essentially measures the average of the squared errors, emphasizing larger errors due to the squaring operation. As a result, MSE penalizes predictions that deviate significantly from the actual values more heavily. One of the key advantages of MSE is that it provides a clear, non-negative measure of model performance. A lower MSE indicates a better fit between the model's predictions and the actual data, while a higher MSE suggests a poorer fit. However, it's essential to keep in mind that MSE is sensitive to outliers because the squared differences can exaggerate the impact of extreme values. In cases where outliers are prevalent, alternative metrics like Mean Absolute Error (MAE) or Huber loss may be more appropriate, as they are less affected by extreme values.

Nonetheless, MSE remains a valuable tool for assessing and comparing the performance of regression models in a wide range of applications.

For this research, we trained and tested 5 models:

* AlexNet [19]
* VGG16 [14]
* DeepPhurie (Proposed)
* Basic CNN [8]
* Hybrid (Proposed)

Various charts based on MSE values were made and evaluated. We will be analyzing the model performances one by one below:

**4.1 ALEXNET**

The chart in figure 7, shows how Alexnet model performed during each fold as the epochs proceeded. We trained the model for 100 epochs over 3 folds, with a batch size of 64. We can observe that at around the 25th epoch, the MSE remained stagnant for validation data over all the 3 folds, but while training this happened much later, at around epoch 40. We can see that for the validation set, the MSE score for the 1st fold is around 210, the MSE score for the 2nd fold is around 260, and for the final fold, the MSE score is around 360.

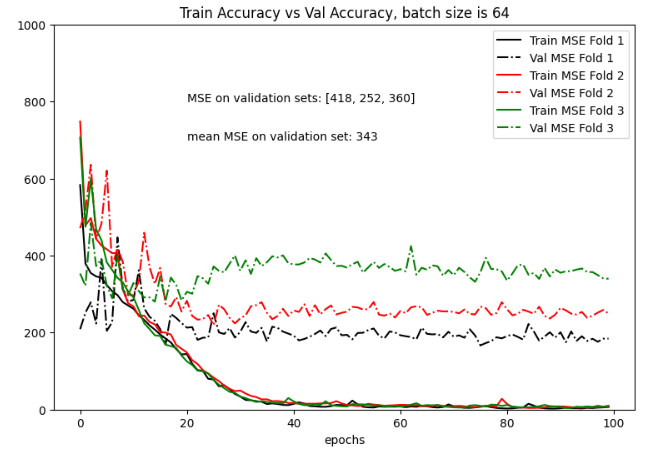


Figure 7: Train accuracy vs Validation accuracy for AlexNet Model

Similarly, The chart in figure 8 shows the model loss for Alexnet model when it is trained

for 100 epochs over 3 folds, with a batch size of 64. This graph shows that the training loss

became stagnant at around the 40th epoch, whereas the validation loss became stagnant much

earlier, around the 25th epoch. We observe that the distance between the training and validation lines increases after the 20th epoch. This shows that the model performance is not up to the mark. This could have happened due to a number of factors like:

* + - Overfitting: It occurs when a model learns the training data too well, capturing noise and irrelevant details. This leads to poor generalization to new data, resulting in low test accuracy. To mitigate overfitting, techniques like regularization, cross-validation, and using a more complex model structure can be employed.
    - Hyperparameter Tuning: The choice of hyperparameters, such as learning rate, batch size, or model architecture, can significantly impact model accuracy. Poorly tuned hyperparameters can cause the model to converge slowly or not converge at all, resulting in suboptimal performance.
    - Model Complexity: Using overly complex models when simpler models would suffice can lead to poor accuracy. A complex model might require more data for training and be prone to overfitting. Selecting an appropriate model complexity is essential for achieving a balance between underfitting and overfitting.
    - Data Leakage: Data leakage occurs when the model inadvertently learns information from the test set during training. This can lead to artificially high training accuracy but reduced test accuracy. Careful separation of training and test data, and feature engineering practices, can prevent data leakage.

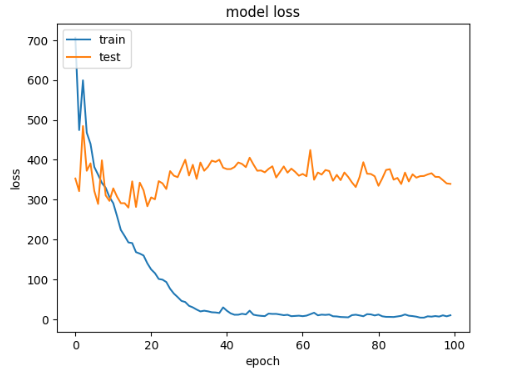


Figure 8: Loss vs Epoch for AlexNet Model, to compare both training & testing phase

**4.2 VGG16**

We trained the VGG16 model for 100 epochs over 5 k-folds, for a batch size of 16. Figure 9 shows the results after training and testing our data on VGG16 model. Based on the graph, the training loss (train) appears to decrease steadily over the course of 100 epochs, which is a positive sign. This suggests that the model is learning the patterns in the training data. The test loss (test) also appears to decrease over time, which suggests that the model is generalizing well to unseen data.

However, it is important to note that the test loss is consistently higher than the training loss. This is a common occurrence in machine learning, and it is known as the generalization gap. The generalization gap occurs because the model is able to memorize the training data to some extent, but it may not be able to generalize this knowledge to new data.

Here are some additional inferences that can be made from the graph:

* The rate of decrease in the training loss appears to slow down after around 50 epochs. This suggests that the model may be starting to approach its convergence point.
* There is some variability in the test loss curve. This variability could be due to the fact that the test loss is calculated on a smaller dataset than the training loss.

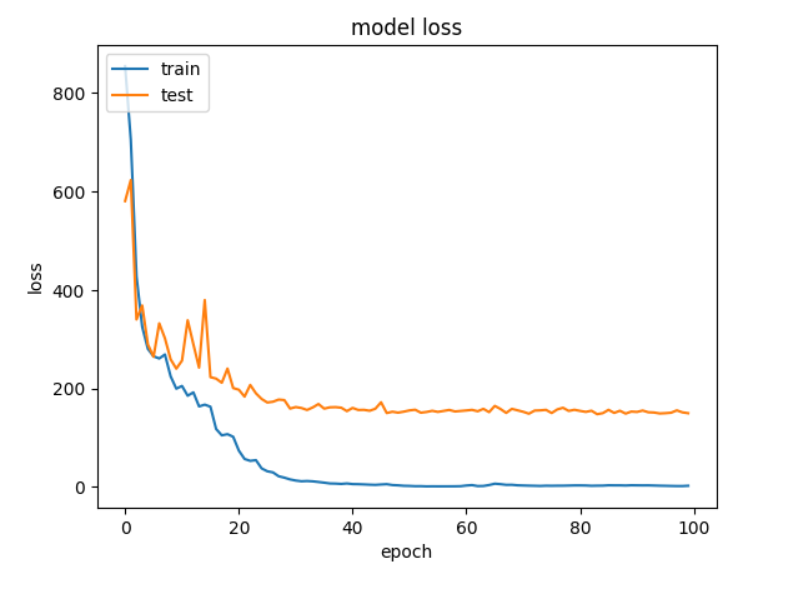


Figure 9: Loss vs Epoch for VGG16 Model, to compare both training & testing phase

Following are the model loss graphs for each fold:

**FOLD 1**

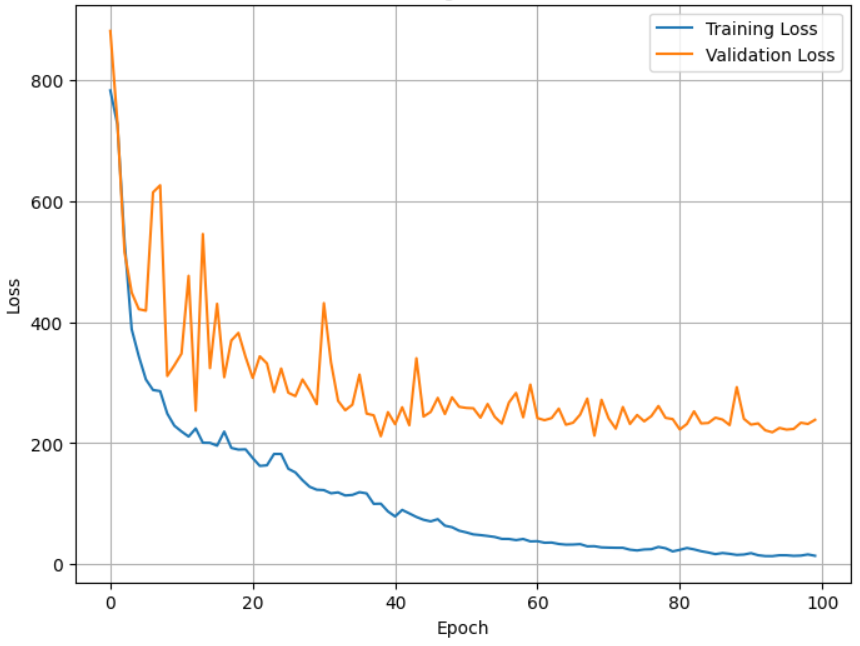
****

Figure 10: Loss vs Epoch for VGG16 Model, Fold 1

**FOLD 2**

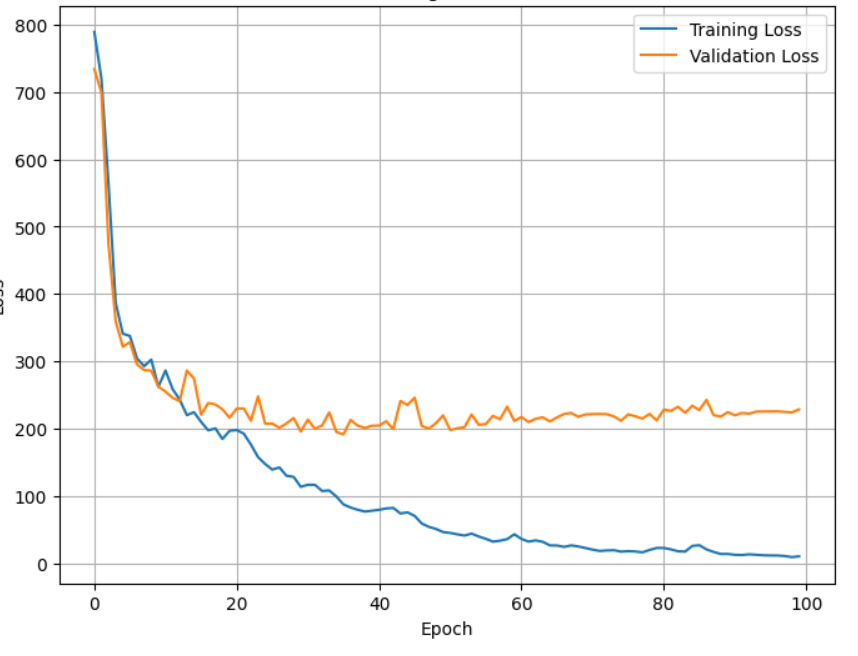
****

Figure 11: Loss vs Epoch for VGG16 Model, Fold 2

**FOLD 3**

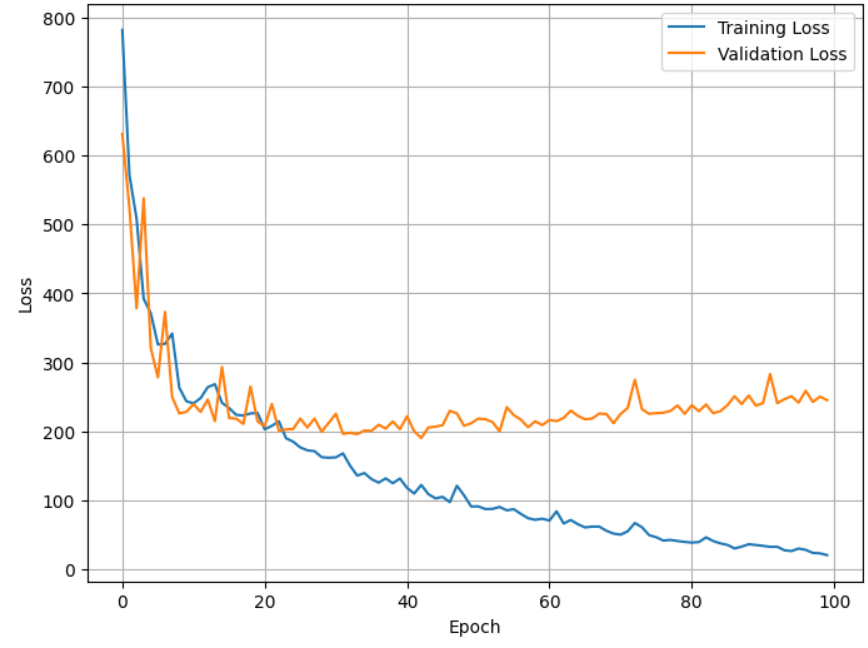
****

Figure 12: Loss vs Epoch for VGG16 Model, Fold 3

**FOLD 4**

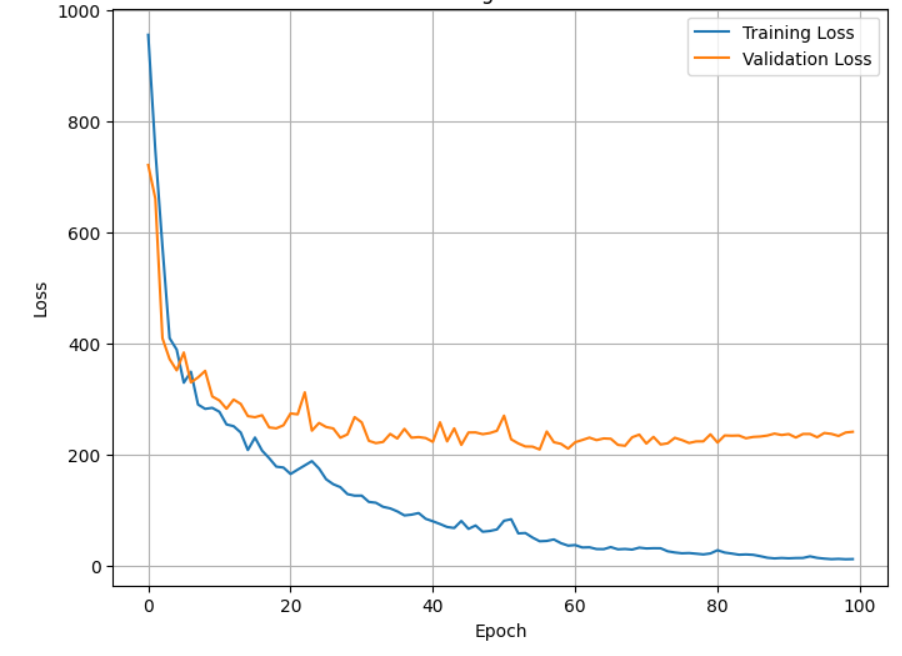
****

Figure 13: Loss vs Epoch for VGG16 Model, Fold 4

**FOLD 5**

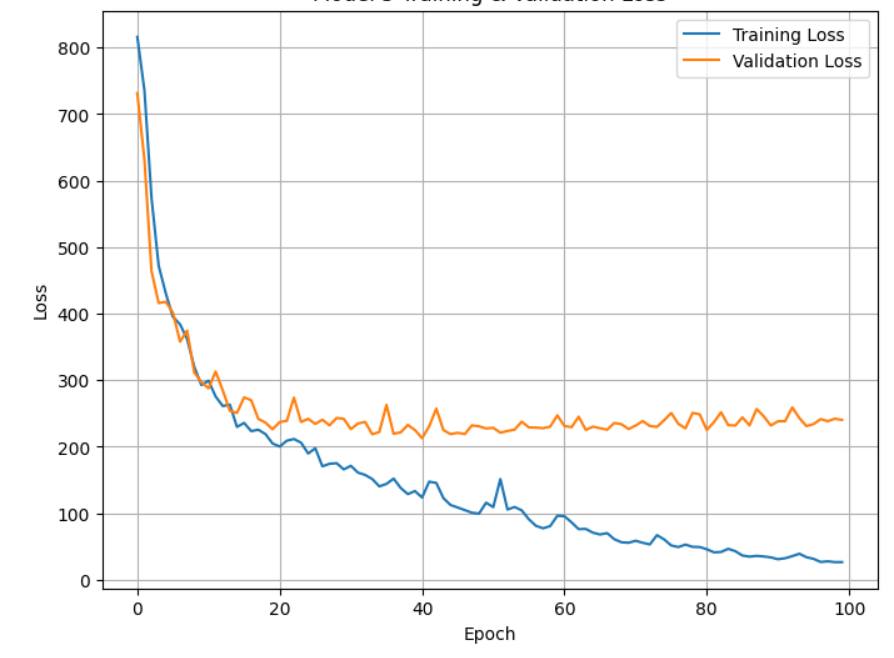


Figure 14: Loss vs Epoch for VGG16 Model, Fold 5

We can see that in the first fold, the validation loss is a little higher with huge spikes in between the initial epochs, but it stabilizes in the next 4 folds. The training loss remains lower only during each fold.

**4.3 HYBRID MODEL**

Figure 15 shows the results after training and testing our data on Mixed Model. It was observed that compared to alexnet and vgg16, this model showed good results. MSE values was constantly low for train, except some spikes at 7-8 intervals during testing phase. There could be many reasons for such improvements in performance like:

* + Contextual Understanding: Multimodal models excel at understanding the context of images and text. They can simultaneously process both visual and textual information, allowing them to grasp the nuances and relationships between the two modalities. This contextual understanding is crucial for tasks like image captioning, where the description should be coherent with the visual content, something that traditional CNNs like AlexNet and VGG16 lack.
  + Semantic Understanding: Multimodal models capture the semantic meaning of the content, allowing them to go beyond simple feature extraction. This semantic understanding enables them to handle complex tasks, such as visual question answering, where an AI system needs to comprehend the question and provide meaningful answers based on the image content.
  + Multimodal Fusion: Multimodal models offer various fusion techniques to combine visual and textual information effectively, which enable them to exploit the complementary aspects of both modalities, resulting in better overall performance. In contrast, traditional CNNs like AlexNet and VGG16 treat images in isolation and lack this fusion capability.

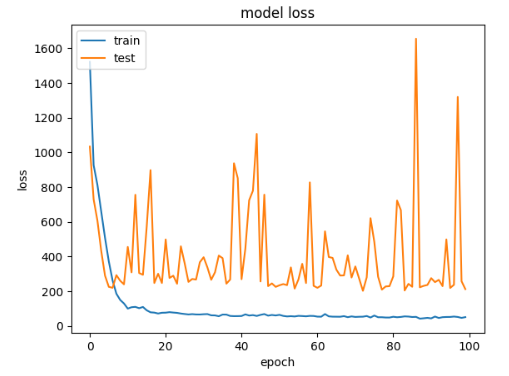


Figure 15: Loss vs Epoch for Hybrid Model, to compare both training & testing phase

**4.4 DEEPPHURIE MODEL**

Figure 16 and 17 shows the results after training and testing our data on DeepPhurie Model. It

was observed that compared to alexnet and vgg16, this model showed good results, similar to

that of the Hybrid model. MSE values were constantly low for train, except some spikes at several

intervals during testing phase. Figure 16 shows that the MSE was high for the first fold initially

and became constant after around 15 epochs and was almost constant for the remaining 4 folds,

while MSE during testing was mostly constant with a few spikes in between.

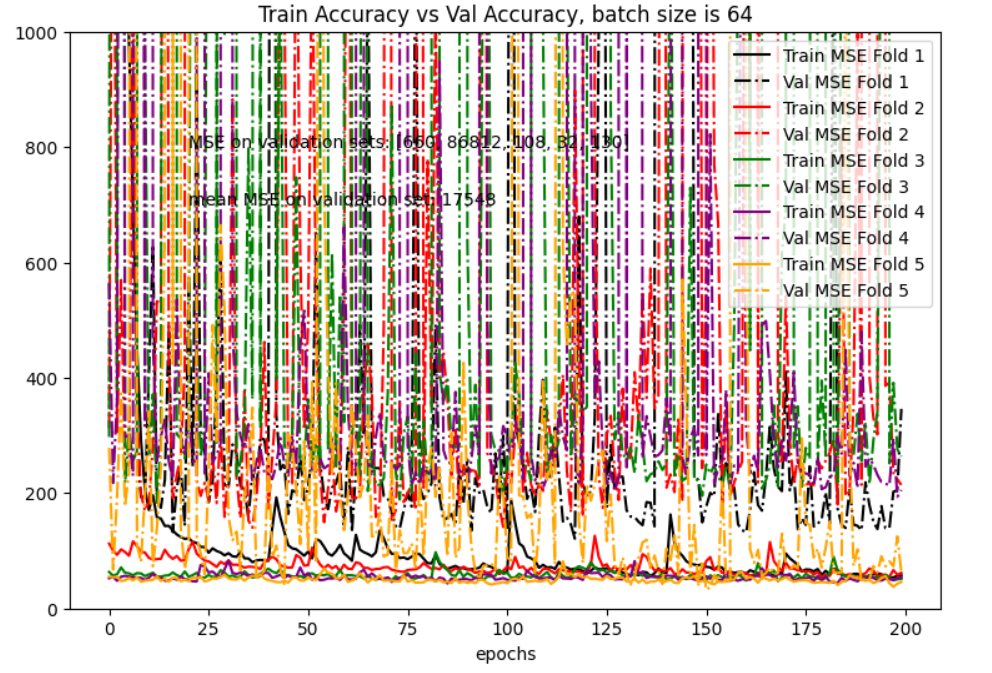


Figure 16: Train accuracy vs Validation accuracy for Deepphurie Model

Figure 17 shows the model loss graph of the Deepphurie model. The training loss was

constantly low from the start till the end whereas during testing, there were a few spikes

in between, around the 20th epoch and somewhere arounf the 162nd epoch, but overall

the testing lines were constantly low. This model showed great results and was one of

the most efficient models in our research.

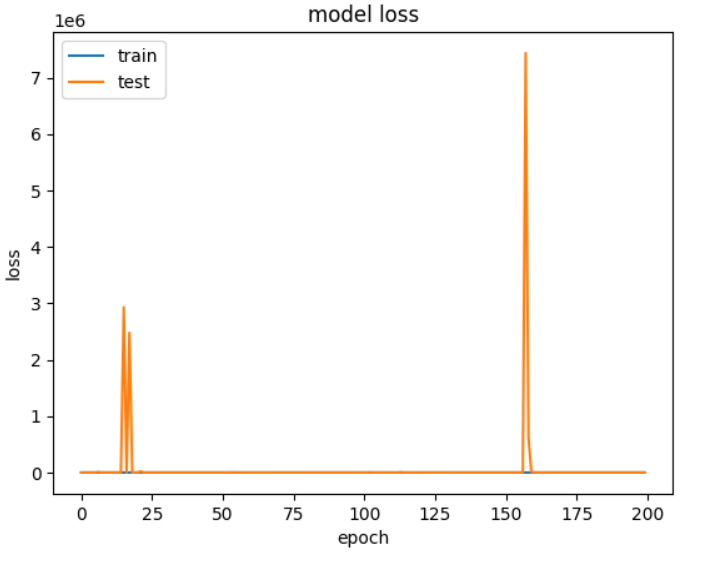


Figure 17: Loss vs Epoch for Deepphurie Model, to compare both training & testing phase

**4.5 BASIC CNN**

Figure 18 shows the results after training and testing our data on Basic CNN Model. It was observed

that compared to all the above used models, like alexnet, deepphurie, hybrid and vgg16, this model

showed good results. MSE value was constantly low for both train and test, except some small

spikes at around 4th or 5th epoch during testing phase. The model was trained at 100 epochs with a

batch size of 16.

The CNN model might have performed better because of multiple reasons:

* **Less Complexity:** As the CNN model is less complex than all the other models used above

we can infer that the MSE for CNN is lower than other models. Increasing

the complexity of the model may lead to problems like overfitting or

underfitting where it becomes difficult for the model to identify and extract

the features for image processing and might lead to poor results.

* **Dataset:** As Alextnet, VGG and Deepphurie are complex models, they might work well for

larger datasets, whereas CNN might work well for smaller datasets.

Here are some additional inferences that can be made from the graph:

* The rate of decrease in the training loss appears to slow down after around 50 epochs. This suggests that the model may be starting to approach its convergence point.
* There is some variability in the test loss curve. This variability could be due to the fact that the test loss is calculated on a smaller dataset than the training loss.

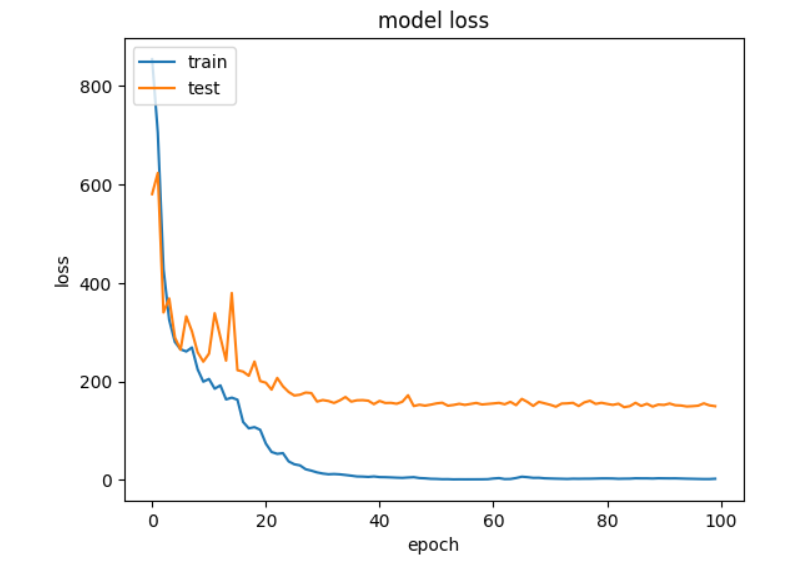


Figure 18: Loss vs Epoch for CNN Model, to compare both training & testing phase

In the below table 1 we have shown various details about our models and our evaluation on how they performed compared to each other.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Aspect** | **AlexNet**  **[19]** | **VGG16**  **[14]** | **DeepPhurie (Proposed)** | **CNN**  **[8]** | **Hybrid**  **(Proposed)** |
| Final MSE Value | 360.024 | 237.38 | 130.93 | 156.69 | 212.5 |
| Number of Layers | 8 | 16 | 22 | 10 | 7 |
| Complexity | Medium | High | High | Low | Medium |
| Evalutaion | Good | Bad | Good | Good | Best |

Table 1: Model Comparisions

# **5. CONCLUSION**

Our findings indicate that traditional CNN architectures, AlexNet and VGG16, while providing a foundation for image-based predictions, exhibit limitations in effectively leveraging numerical data. The novel model, which combines image and numerical data, has shown promise in significantly improving the accuracy of cyclone intensity predictions. The ability to incorporate both visual and numerical information has allowed the model to capture a more comprehensive understanding of the complex factors influencing cyclone intensity.

Furthermore, the research has underlined the importance of multimodal models, such as the one we proposed, in tackling multidisciplinary problems like cyclone intensity prediction. By fusing information from diverse sources, these models can harness the complementary aspects of each data type and provide a more holistic and contextually rich prediction, leading to more reliable forecasts.

Our study has made significant development in cyclone intensity prediction using deep learning, and it strongly advocates the incorporation of numerical data alongside image data for more accurate and comprehensive forecasts. This research paves the way for cyclone prediction methodologies and underscores the potential of deep learning models to contribute to our understanding and mitigation of natural disasters.

# **6. FUTURE SCOPE**

The research presented in this paper has established the potential of deep learning models for cyclone intensity prediction. However, there remain several exciting avenues for future research and improvement in this field:

1. **Data Augmentation and Transfer Learning:** Future research can explore data augmentation techniques specific to cyclone imagery to further enhance the robustness of models. Additionally, leveraging transfer learning from general meteorological datasets and applying it to the task of cyclone intensity prediction can improve model performance and generalization.
2. **Explainable AI (XAI) Techniques:** Developing interpretable models is crucial in the domain of cyclone intensity prediction, as it can provide insights into the features and patterns that contribute to the predictions. The integration of XAI techniques can help in understanding the deep learning models' decision-making processes, enhancing their transparency and trustworthiness.
3. **Real-Time Deployment:** To make cyclone intensity prediction more practical, the future research should focus on real-time deployment. Developing systems that can provide continuous monitoring and prediction of cyclone intensity in a live operational environment is crucial for disaster preparedness and response.
4. **Climate Change Adaptation:** Considering the increasing impact of climate change on cyclone behavior, future research can delve into understanding and modeling the interplay between climate change factors and cyclone intensity. This can help in developing models that account for changing environmental conditions.
5. **Integration with Early Warning Systems:** Future work should focus on integrating deep learning-based cyclone intensity prediction models with existing early warning systems to provide timely and accurate information to at-risk communities and authorities.