CONSTRUCTOR UNIVERSITY

Weather Classification using Convolutional Neural Networks

by

Mohamed Shaaban

Bachelor Thesis in Computer Science

Submission: May 16, 2023

Supervisor: Dr. Fangning Hu

Constructor University | School of Computer Science and Engineering

Statutory Declaration

Family Name, Given/First Name	Shaaban, Mohamed
Matriculation number	30002592
Kind of thesis submitted	Bachelor Thesis

English: Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

This document was neither presented to any other examination board nor has it been published.

German: Erklärung der Autorenschaft (Urheberschaft)

Ich erkläre hiermit, dass die vorliegende Arbeit ohne fremde Hilfe ausschließlich von mir erstellt und geschrieben worden ist. Jedwede verwendeten Quellen, direkter oder indirekter Art, sind als solche kenntlich gemacht worden. Mir ist die Tatsache bewusst, dass der Inhalt der Thesis in digitaler Form geprüft werden kann im Hinblick darauf, ob es sich ganz oder in Teilen um ein Plagiat handelt. Ich bin damit einverstanden, dass meine Arbeit in einer Datenbank eingegeben werden kann, um mit bereits bestehenden Quellen verglichen zu werden und dort auch verbleibt, um mit zukünftigen Arbeiten verglichen werden zu können. Dies berechtigt jedoch nicht zur Verwendung oder Vervielfältigung.

Diese Arbeit wurde noch keiner anderen Prüfungsbehörde vorgelegt noch wurde sie bisher veröffentlicht.

.....

Date, Signature

Abstract

The field of machine learning and computer vision has quickly advanced in recent years, and It is becoming increasingly relevant in our daily lives. As such, there is a growing need for individuals to understand the workings of these technologies and how they can be leveraged to our advantage. This paper aims to provide more detail on convolutional neural networks for the problem of weather classification. The reason for choosing weather classification is due to the many disastrous events caused by adverse weather that have repeatedly occured all across the globe. Another reason for choosing this problem is that it affects all of us on daily basis. Especially with the weather changing multiple times in the same day, having more accurate weather prediction can help us adjust accordingly.

The main question here is ”Can machine learning algorithms accurately classify weather images in real-world scenarios?” as well as ”How accurate can the model be?”.

These questions were addressed by discussing relevant work and theory as well as multiple hands-on experiments to find good models and improve on them. In an attempt to create a model from scratch by stacking layers and seeing the results, the model started with around 50 percent accuracy and was only able to reach 81 percent on validation data and couldn't do much improvements. Transfer Learning was also used and Resnet152V2 was found to be the better model on this data reaching over 86 percent.

In an attempt to demonstrate how accurate the model can be, visualizations were created to show sample images and the model's prediction for them. Similarly, out-of-distribution data was used to show how the model performs on other data, some of those images were taken from my university's campus showing different weather conditions or meteorological phenomenon. The results show good predictions and provide an idea on how similar models can improve further.

Contents

1 Introduction

1.1 Motivation

Unfortunate incidents due to bad weather have occured time and time again all around the globe. Some of the tragic events that have recently made news include airline crashes, railway derailments, ship collisions, and automobile accidents. The society has become aware of this serious issue of safety and security in dangerous situations, and many studies have been conducted in the past to highlight how vulnerable transportation services are due to disruptions caused by weather. Weather-controlled driving behavior and speeds have been suggested in the past. Automated weather forecasting has increasing importance now that intelligent transportation is a growing subject with technology expanding day after day. [5]

Massive numerical simulation systems are heavily used in modern weather prediction. Those of which are frequently run by meteorological agencies all around the world. Even though millions of different measurements are regularly taken, these observations are still insufficient to fully characterize the condition of the planet's atmosphere and other parts of the Earth's system which it interacts and exchanges mass or energy with. Thus, these numerical simulations are then aided with deep learning and weather classification for better results. [15]

The weather has a big impact on us in our daily lives through the solar energy system and outdoor sporting events, for example (by heavy rain, haze, etc.). Therefore, impacting the functioning of many visual systems, such as outdoor video surveillance and vehicle assisted driving systems. There is little doubt that many visual and meteorological systems depend on the ability to categorize the weather from a single image, sometimes known as the "weather classification problem." To categorize the weather nowadays, costly sensors or human eyes are usually utilized. Due to a lack of essential staff and/or expensive equipment, the ability to measure local weather conditions is limited since weather conditions are local. Recently, researchers proposed that computer vision algorithms may be created to precisely categorize weather conditions using photos since inexpensive security cameras are widely available and would be adequate to assess weather. This may save money on expensive human and technical resources like sensors. [3]

1.2 Relevant work

Weather classification from images is a complex job in computer vision due to the diverse as well as the intricate characteristics of weather images. Conventional machine learning methods require accurate image segmentation and pre-processing to extract relevant features, which can be challenging to perform in practice. In contrast, convolutional neural networks (CNNs) have emerged as a powerful approach for weather phenomenon recognition, obtaining great accuracy without the requirement for laborious pre-processing or manual feature extraction.

However, training more complex CNNs with hundreds of millions of model parameters requires a substantial amount of data to avoid overfitting. Transfer learning has been used to alleviate this issue by fine-tuning pre-trained models on other datasets, but deploying such a large model can be complicated. Moreover, transfer learning limits the flexibility of the model's design, making it challenging to tailor the model to specific data.

Alternatively, lightweight CNNs have been developed to reduce redundancy and compu-

tational consumption while maintaining high accuracy. One popular approach is the 1x1 convolution module, which reduces the amount of model parameters. Another method is depth-wise convolution, which is now a fundamental component of lightweight networks. Other techniques, such as pruning, quantization, and knowledge distillation, have also been used to reduce network redundancy.

The graph below shows a comparison between different models and their Macro-Precision, Macro-Recall, Macro-F1 and Accuracy.

Table 2
Comparison With Other Models by Accuracy and Macro-Average of Precision, Recall, and F_1-Measure

Model name	macro_ Precision	macro_ Recall	macro_F1	Accuracy
MeteCNN	0.9355	0.9331	0.9340	0.9268
Vgg16	0.8776	0.8750	0.8750	0.8712
Vgg19	0.8531	0.8555	0.8531	0.8521
Resnet18	0.8868	0.8953	0.8901	0.8873
Resnet34	0.8932	0.8955	0.8928	8,858
Efficientnet-B7	0.8819	0.8804	0.8805	0.8741
MobilenetV3-small	0.8507	0.8413	0.8420	0.8404
MobilenetV3-large	0.8770	0.8749	0.8751	0.8712
Wang et al. (2020)	0.8604	0.8522	0.8546	0.8521

Abbreviation: CNN, convolutional neural network.

Figure 1: 'A comparison between different models for weather classification' [21]

To understand the comparison better, a deeper understanding of macro-precision, macro-recall, macro-F1, and accuracy (which are all evaluation metrics) would be helpful.

Macro-precision:

The proportion of genuine positive occurrences to the total number of positive examples predicted is known as precision. The classification task's macro-precision is the arithmetic mean of precision for all classes. By adding together the accuracy scores for each class and dividing by the total number of courses, it is computed. When the dataset is unbalanced, macro-precision is a beneficial metric to employ since it assures that each class is given the same weight in the computation.

Macro-recall:

The proportion of genuine positive occurrences to all instances in the positive class is referred to as the recall rate. The term ”macro-recall” refers to the classification task's arithmetic mean of recall across all classes. By adding together the recall scores for each class and dividing by the total number of classes, it is computed. When the dataset is unbalanced, Macro-Recall is a valuable metric to employ since it assures that each class is given the same weight in the computation.

Macro-F1:

The harmonic mean of recall and accuracy is the F1 score. It balances the two parameters by taking into consideration both accuracy and recall. Macro-F1 is the classification task's arithmetic mean of F1 scores for all classes. It is computed by adding together the F1 scores for each class and dividing by the total number of classes. The macro-F1 metric can be used when the dataset is unbalanced since it gives a single number that captures the performance of the model as a whole.

Accuracy:

The proportion of overall correct forecasts to all of the model's predictions is referred to as accuracy. It is a simple statistic that is widely used to evaluate the effectiveness of classification models. However, if the dataset is heavily biased towards one class, accuracy metric might not be appropriate because it can give a false impression of the performance.

In this paper, the intention is to develop a convolutional neural network with the aim of achieving a similar objective to that of the previous studies mentioned.

[8]

2 Theory

2.1 Artificial Neural Networks

The operation of organic nervous systems, such as the human brain, is a major source of inspiration for artificial neural networks (ANNs), which are computer processing systems. The primary building block of ANNs is a large number of linked computational nodes (also known as neurons), which collaborate in a dispersed manner to learn from the input and optimize the final output.

A model of an ANN's fundamental composition may be seen in the following figure 2. The input would be loaded into the input layer, which would then distribute it to the hidden layers. The input is typically in the form of a multidimensional vector. The learning process is when the hidden layers consider judgments from the preceding layer and determine if a stochastic change inside itself improves or worsens the output. Deep learning is a term used to describe the stacking of many hidden layers. [12]

Input layer: Input nodes are often referred to as the information from external sources that are provided to the model to learn from and make conclusions from. The subsequent layer, the hidden layer, receives data from the input nodes.

Hidden layer: All calculations on the input data are performed by the group of neurons in the hidden layer. Any number of hidden layers are possible for neural networks. The simplest network has a single hidden layer.

Output layer: The findings and output of the model are the results of all calculations. The output layer may have one or more nodes. In contrast to binary classification problems, where the output node is always 1, multi-class classification problems may have more than one output node. [2]

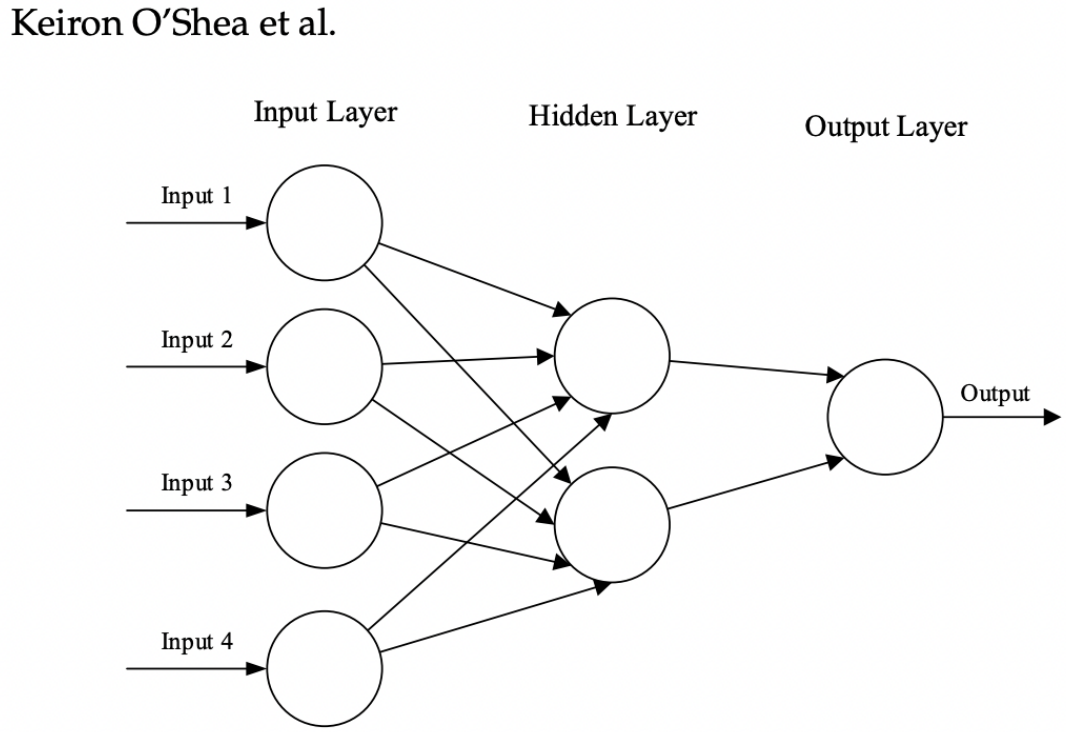

Figure 2: 'example of an ANN'

[12]

2.2 Convolution

The convolution operation involves sliding a small filter, often referred to as a kernel or a window, over the entire input image. The kernel multiplies the pixel values of the inputted image within its receptive field and sums up the results, which produces a single output value. This process is repeated for all positions on the input image, resulting in a new feature map highlighting specific patterns or structures in the inputted image. The figure 3 below shows an example of the convolution operation.

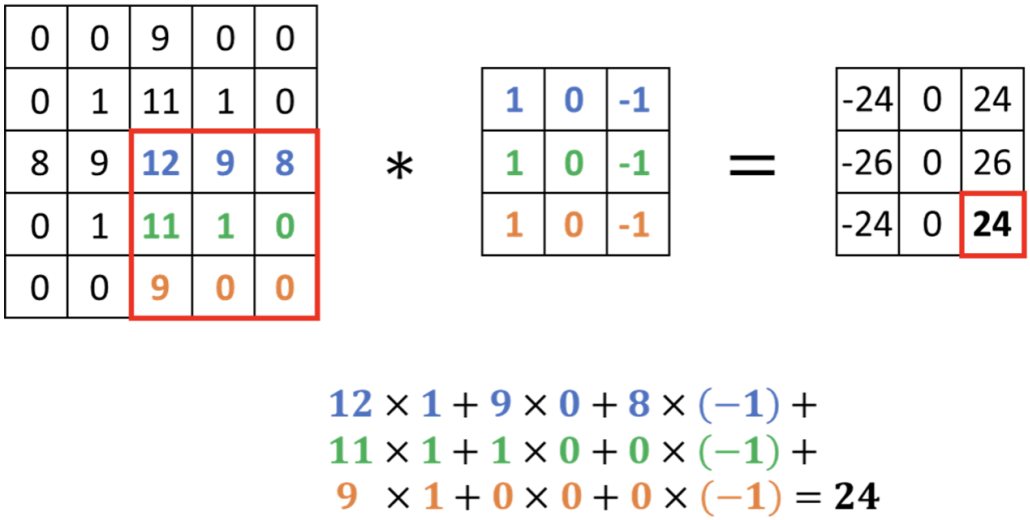

Figure 3: 'Convolution to detect vertical edges'

[17]

2.3 Convolutional Neural Networks

Convolutional Neural Networks are constructed by combining the use of ANN with Convolution. The Neural Networks consists of an input layer, convolution layers, polling layers,and fully-connected layers. Figure 4 shows an example.

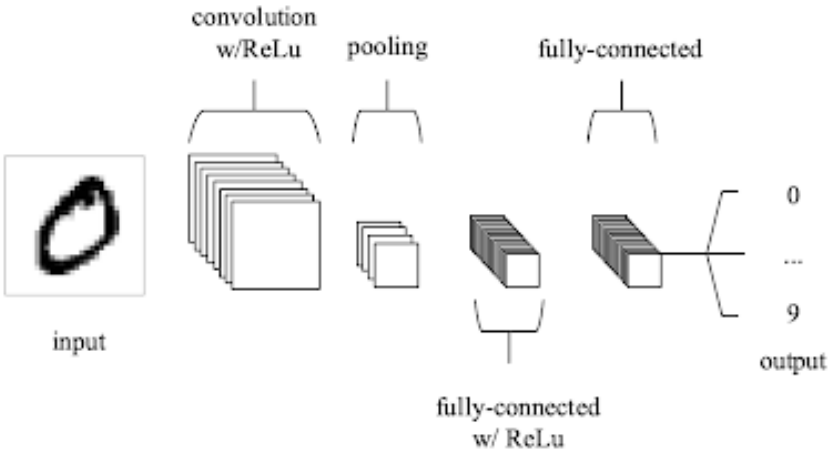

Figure 4: 'Example of a simple Convolutional Network to for detecting an image'

[12]

2.3.1 Input layer

The input layer is responsible for receiving the raw pixel values of the input image, which are commonly represented as a vector of pixels. Each pixel represents either the colour

or the intensity. For example, a grayscale image with dimensions 28x28 pixels would have 784 values, each of which can range from 0 to 255. 0 illustrates black and 255 illustrates white.

The input layer neurons receive these pixel values and passes them on to the neurons in the next layer of the network. Commonly, prepossessing operations such as scaling or normalization can also be performed on the input image to ensure that the pixel values are in a suitable range for the subsequent layers to work with.

The size of the input image is often what determines how many neurons are present in the input layer. For example, an image with dimensions 28x28 pixels would require an input layer with 784 neurons, while a larger image with dimensions 224x224 pixels would require an input layer with 50,176 neurons.

Overall, the input layer plays a critical role in image recognition using neural networks, as it is responsible for receiving and processing the raw pixel values of the input image and passing them on to the subsequent layers for further processing and analysis. [12]

2.3.2 Convolution layer

Convolution is one of the most fundamental components for convolutional neural networks (CNNs or ConvNets). it learns from the input pictures and recovers their characteristics. The aim of the convolution operation is to extract important properties of the input picture, such as edges, curves, and forms that can then lead to extracting high-level features such as a face.

ConvNets typically consist of multiple convolutional layers, where each layer gives a collection of learnable filters to the inputted image. The filters are applied to small patches of the input image, resulting in a feature map that records the place in the input image where the filters was activated. These feature maps can be thought of as maps of the presence of certain features, such as edges, textures, and shapes.

The filters get more complicated and specialized as the network goes through more convolutional layers, and the feature maps start to capture more and more high-level characteristics. For instance, in a network created to recognize faces, the initial convolutional layer would identify fundamental characteristics like edges and curves, while further layers might identify facial characteristics like eyes, noses, and mouths, and finally identify the entire face.

The ability of ConvNets to learn and extract high-level features from input images enables them to comprehend the dataset's pictures holistically in a way that is comparable to how humans do. ConvNets are therefore especially useful for tasks like image segmentation, object detection, and classification, where the network needs to understand the content of the image and identify the objects and their boundaries. [12]

2.3.3 Pooling Layer

Convolutional neural networks (CNNs) frequently employ the pooling operation to decrease the spatial dimensions (width and height) of feature maps while maintaining the depth dimension. Pooling's main objective is to cut down on the amount of parameters in the network, which can aid in preventing overfitting and accelerating training.

Different pooling operations exist, including L2 pooling, average pooling, and max pooling. The most popular pooling method is called max pooling, which chooses the highest value from a window (often 2x2 or 3x3) and discards the rest. On the other hand, average pooling determines the window's average value and discards the remainder.

Pooling can be an efficient technique to minimize the dimensionality of the feature maps, but if applied incorrectly, it can also be detrimental and diminish output accuracy. This is so that information that could be valuable for the intended purpose isn't lost during pooling. For instance, pooling in object identification may result in the network losing spatial knowledge about object borders, making it more challenging to correctly categorize things.

Several methods have been put up to alleviate the drawbacks of pooling in order to solve this issue. Fractional max pooling, one of these methods, executes pooling in a stochastic manner, enabling a more fine-grained representation of the feature maps. A different method is known as global average pooling, which executes pooling across the whole feature map and generates a single value for each feature map while, to some extent, conserving spatial information. [12]

2.3.4 Fully-connected layers

Each neuron in a layer of a neural network that is fully-connected, sometimes referred to as dense layers, is coupled to every neuron in the layer above it. To put it another way, each neuron in a fully linked layer receives input from every node in the layer above and contributes to every neuron in the layer below. Fully linked layers become more expressive as a result, but they are also computationally expensive and subject to overfitting.

A fully-connected layer's weights and biases are learned during training using backpropagation. A weighted total of the inputs from the layer before is added along with a bias term to determine each neuron's output in the layer. During training, the weights and biases are modified to reduce the loss function and raise the network's accuracy.

Deep learning architectures frequently employ fully-connected layers, particularly for image classification applications. They are generally employed to carry out the final classification in the network's last few levels.

Fully connected layers have a significant computational cost, particularly for big input data sets. This is due to the fact that the layer's number of connections expands quadratically with the number of input neurons. Their inclination for overfitting is another drawback, as the layer's numerous parameters may encourage the network to memorize training data rather than discover broader patterns.

Numerous methods have been suggested to overcome these problems, including regularization, which adds a penalty term to the loss function to deter excessive weights, and dropout, which randomly removes some neurons during training to prevent overfitting. Convolutional layers or recurrent layers may also be used in place of fully-connected layers, depending on the task at hand and the nature of the input data. [12]

2.4 Transfer Learning

An initial large dataset is used to train a transfer learning model, and a subsequent smaller dataset is used to refine it. Transfer learning's primary objective is to use the knowledge

learned from the large dataset to perform better on the smaller dataset.

In classical machine learning, the model is created from scratch and trained on a particular dataset, which is usually timely and computationally expensive. Transfer learning helps to minimize the amount of training needed for the target task because the pretrained model already has some prior knowledge. It is especially beneficial when the target dataset is small or the data is comparable to that of the trained model.

Transfer learning is frequently used in several disciplines, including speech recognition, natural language processing, and computer vision, and has shown to be very successful in enhancing model performance. It is a powerful method that enables quicker and more effective model training and has the ability to raise model accuracy even when the target dataset is small.

[19]

2.5 Overfitting and Underfitting

Overfitting and underfitting are common challenges in machine learning, particularly when training complex models like deep neural networks.

An overfitted model performs poorly on fresh data because it is over complicated and matches the training data too closely. This can occur when the model has too many parameters or when the training data is excessively small compared to the model's complexity. Overfitting leads to poor generalization performance, where the model performs well when using training data but poorly with newly acquired information.

To avoid overfitting, several techniques can be used, including regularization, early stopping, dropout, and data augmentation. Regularization imposes a penalty expression to the loss function to prevent the model from overfitting. Early stopping halts the training when the validation loss stops improving. Dropout randomly excludes some neurons during training. Data augmentation artificially generates more training data by applying transformations such as rotation, scaling, or flipping.

When a model is too simplistic to detect the important patterns in the data, underfitting occurs, which results in subpar performance on both training and test data. To address underfitting, the complexity of the model can be increased, or more data can be collected, or the data can be preprocessed to extract more relevant features.

It is essential to balance the complexity of the model given the volume of available data to achieve the best generalization performance. Both overfitting and underfitting potentially result in subpar performance on fresh data, and it is crucial to choose appropriate techniques to address them.

The following are possible solutions to try to fix our problem: Early-stopping, Network-reduction, Augmentation of training data.

Early-stopping:

Stopping the model prior to conversion at a point when the model is improving on both validation and training sets (as shown in figure 5). This helps find the best model for both training and testing datasets.

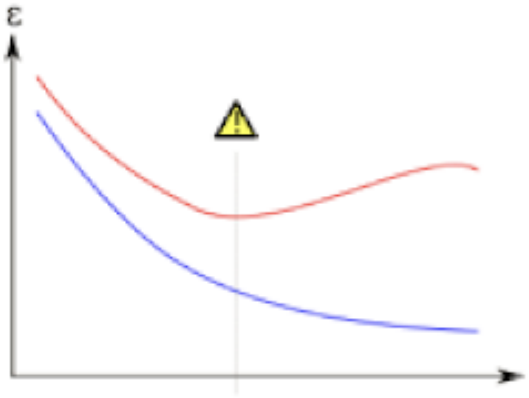

Figure 1. Validation error vs testing error

Figure 5: 'Early stopping here is necessary to reduce overfitting and to reach best testing accuracy'

[12]

Network-reduction:

Eliminating pointless or redundant model parameters can reduce the complexity or size of a neural network using the pruning strategy. Pruning is used to preserve or even increase the model's accuracy on the training data while enhancing the generalization performance and lowering the effect of overfitting.

Augmentation of training data:

Increasing the data can improve our model and reduce the effect of overfitting. The issue is that getting more data is expensive and usually needs human labor for labeling. It is also very difficult to get data without noise.

3 Implementation

3.1 Objective

This project's objective is to create a convolutional neural network (CNN) that can achieve performance similar to the previous work. To achieve this, we train a CNN using datasets relevant to our problem domain and optimize its performance using appropriate architectures and techniques. In particular, we investigate the impact of different CNN architectures, activation functions, pooling strategies, and regularization techniques on model accuracy and generalization. In addition, we investigate how data augmentation and transfer learning can be used to improve model performance on limited datasets. By developing a CNN that can achieve results comparable to previous studies, this work aims to advance the field and demonstrate the potential of deep learning in solving real-world problems.

3.2 Data

The dataset used for this paper includes 6862 images of various weather phenomenon. The pictures are divided into 11 labels: dew, fogsmog, frost, glaze, hail, lightning ,rain, rainbow, rime, sandstorm and snow. [21]

The depicted diagram (figure 6) presents a comprehensive breakdown of the distribution of data points (images) across various classes. In order to effectively work with this dataset, it is imperative to have a clear comprehension of the underlying information and its distribution. To this end, the illustration is accompanied by examples of images and their corresponding class labels, thereby providing valuable insight into the nature of the data. Such information is vital for conducting accurate analysis and making informed decisions based on the data.

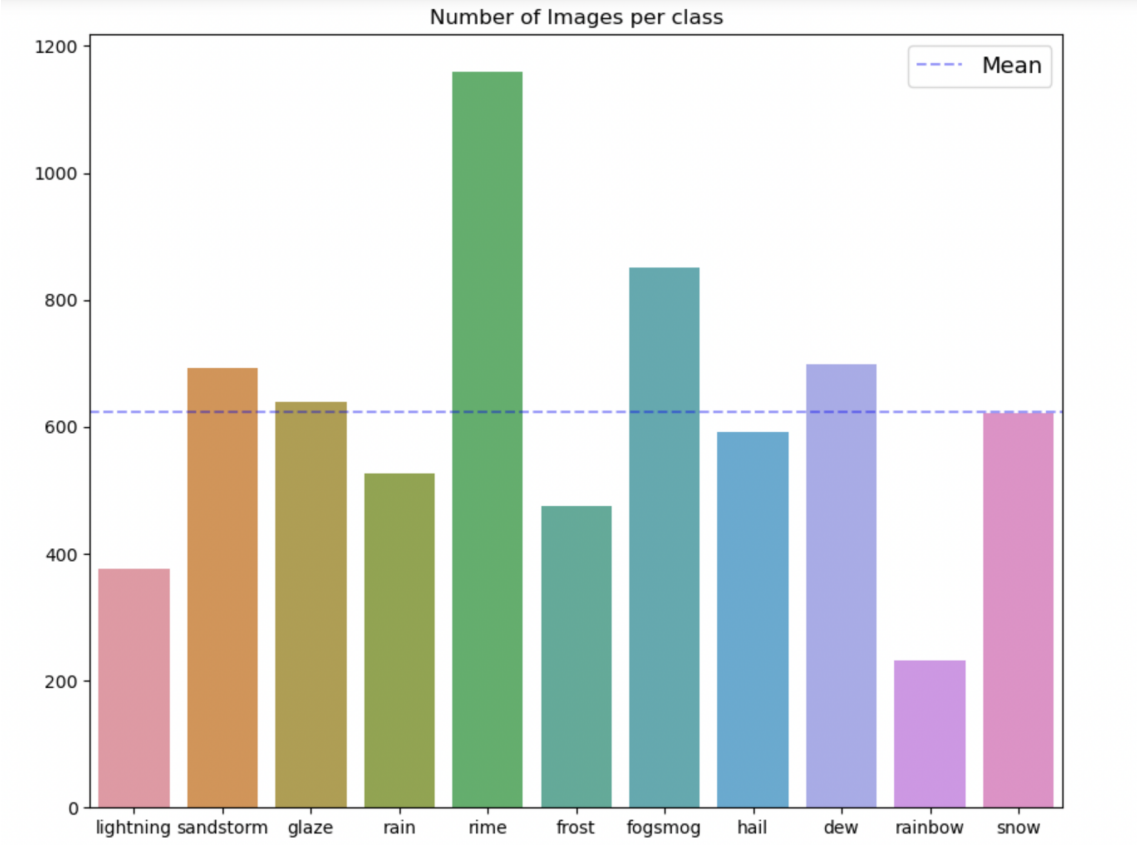

Figure 6: 'the distribution of the data'

The provided visual aid showcases a sample of images that need to be accurately classified. As evident from the image set, the classification task poses certain challenges, primarily due to the potential variability in image quality and the subjective nature of human classification. To overcome these challenges, it is crucial to establish clear criteria for distinguishing between different image classes. Notably, color appears to play a significant role in the classification process, implying that grayscale images might be unsuitable for this particular task. Hence, a systematic and data-driven approach is warranted for achieving reliable and accurate image classification.

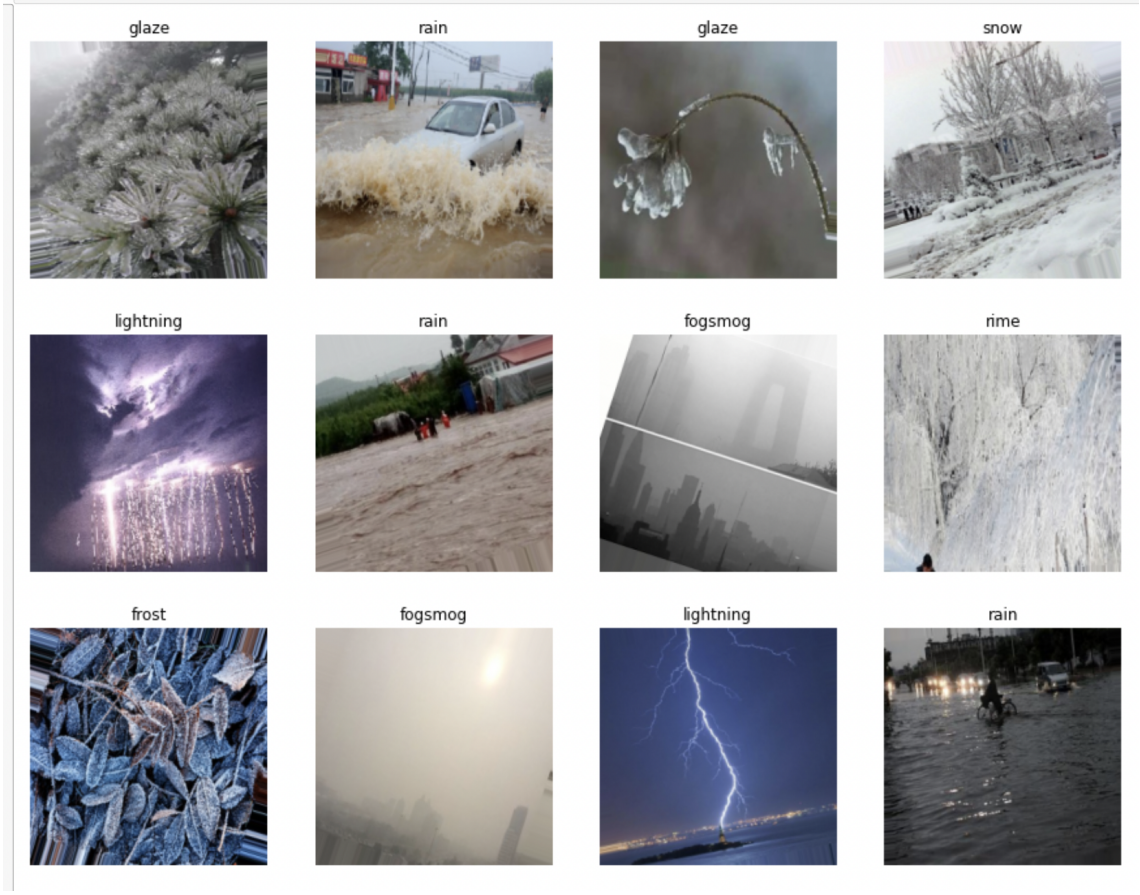

Figure 7: 'visulization of the data'

Dealing with weather data, specifically images, can be challenging for several reasons. The following are some important aspects that make it challenging to divide weather photographs into the 11 main categories such as variability, subjectivity, limited data and noise.

Variability:

The time of day, season, location, and other factors may all have a big impact on the weather. Depending on the circumstances in which they were taken, images of the same weather (such as rain) may appear extremely differently. It is challenging to establish precise standards for grouping photos into different groups because of this unpredictability.

Subjectivity:

Weather is a complex and dynamic phenomenon, and different people may interpret weather images differently based on their own experiences and biases. For example, what one person might classify as fog, another person might classify as smog. This subjectivity can make it challenging to achieve consistent and reliable classifications across different observers.

Limited data:

It may be challenging and expensive to gather a lot of meteorological data, especially for less frequent weather occurrences. Due to skewed datasets and a lack of instances

for some classes, it might be challenging to develop reliable classifiers for those classes.

Noise:

Noise from air distortion, camera artifacts, and other visual oddities can be quite present in weather photographs. It may be challenging to distinguish and categorize various weather patterns effectively due to this noise.

Overall, it is a difficult undertaking that calls for careful analysis of these and other aspects to classify weather photos into 11 unique groups. Robust algorithms that can take into account subjectivity, variability, limited data, noise, and the intricacies of the weather are necessary for the effective categorization of weather photos.

3.3 Model

3.3.1 Library Overview and Functionality

Sequential:

A linear stack of layers in a neural network is represented by the Sequential function class in the Keras library. By adding layers to the stack one at a time in the order you want them to be executed, you can easily build a neural network.

Typically, a feedforward neural network is built using the sequential function, in which the input is fed through each layer in turn to produce an output. Convolutional, activation, pooling, and other types of layers are examples that can be used in the Sequential model.

The Sequential function provides a simple way to create neural networks, particularly for beginners who are just starting out with deep learning. By using the Sequential function, you can quickly and easily create a neural network architecture without having to worry about the details of how the layers are connected or how the data flows through the network. You can simply add layers one by one, and Keras will take care of the rest. [20]

Conv2D:

In a neural network, the Conv2D function is a 2D convolutional layer that applies a number of filters to an input picture. The filters that Conv2D employs are often used for tasks like feature extraction and picture categorization. They are learned during the neural network's training phase.

Convolutional kernels or filters are frequently used to refer to the filters used by Conv2D when discussing convolutional neural networks (CNNs). These filters are generally tiny matrices (like 3x3 or 5x5) that are applied in a sliding window style to specific small areas of the input picture. Each filter is applied to the input image numerous times, with each application shifting the location of the filter inside the image. To create the output of the convolutional layer, the outcomes of these procedures are then added.

The specific properties of the filters that Conv2D learns will depend on the neural network's chosen design and the kind of input data. However, Conv2D's learned filters frequently aim to recognize specific patterns or components in the input image, such as edges, corners, or textures. By using a combination of these filters on the input image, Conv2D may be able to learn to extract higher-level characteristics from the image, relevant to the task at hand (such as object identification or image classification). [20]

MaxPooling2D:

In convolutional neural networks, MaxPooling2D is a type of pooling layer that is commonly used to downsample an input feature map. MaxPooling2D operates over a two-dimensional input, typically an image, and reduces the spatial dimensions (height and width) of the feature map by taking the maximum value in each non-overlapping rectangular subregion of the input.

Mathematically, MaxPooling2D can be defined as follows:

• Given an input feature map of size H x W x C, where H is the height, W is the width, and C is the number of channels.

• Divide the input into non-overlapping rectangular subregions (or windows) of size, pool-size x pool-size, where pool-size is a hyperparameter.

• For each subregion, take the maximum value of the feature map within the subregion.

• The output feature map has spatial dimensions of H/pool-size x W/pool-size x C, where / represents integer division.

MaxPooling2D aids in reducing the feature map's spatial dimensions, which can assist in lowering the amount of parameters used in the network and avoiding overfitting. By taking the largest value in each subregion, it also aids in the extraction of the most important characteristics from the input.

[20]

Dropout:

Dropout is a regularization technique that is commonly used in deep learning to avoid overfitting. Dropout works by randomly "dropping out" (i.e., setting to zero) some of the neurons in a layer during training. This forces the remaining neurons to learn more robust features and reduces the risk of overfitting.

During training, at each update, a Dropout layer randomly sets a portion of the input units to zero. The fraction of units that are set to zero is a hyperparameter and is typically set to a value between 0.2 and 0.5. During testing, the Dropout layer does not change the input.

[20]

Flatten:

Flatten is a function that is used to convert a multidimensional array or tensor into a one-dimensional array. This is typically used as a preprocessing step before passing the data into a dense or fully connected layer in a neural network.

Flattening a tensor is often necessary in deep learning when you want to pass the data through a fully connected layer, which expects a one-dimensional input. For example, when processing images in a convolutional neural network, the output of the convolutional layers is typically a multidimensional tensor, which is then flattened before being passed into a fully connected layer for classification.

[20]

Dense:

Deep learning frequently uses the Keras function, Dense, to build fully-connected neural networks. Each connection has a weight that is learnt during training, and a Dense layer connects every neuron in the input layer to every neuron in the output layer.

The density of the layer's neurons and the activation function that is applied to each neuron's output serve as its defining characteristics. To obtain the optimum performance, the number of neurons is a hyperparameter that must be adjusted for each network design and dataset. The network is given nonlinearity through the activation function, which enables it to recognize intricate patterns in the data.

[20]

ResNet152v2

ResNet152v2 is a modification of the 152-layer original ResNet architecture put forth by Microsoft Research. ResNet152v2, like ResNet50, employs the residual learning method to speed up the training of very deep neural networks.

The "v2" in the name stands for version 2, which includes several improvements over the original ResNet architecture. These improvements include using bottleneck residual blocks with smaller convolutional filters, which aids to reduce the amount of parameters and computational cost while maintaining the same level of accuracy.

On a variety of computer vision operations including classification, object detection, and semantic segmentation, ResNet152v2 has demonstrated impressive performance. It has produced cutting-edge outcomes on well-known benchmark datasets like ImageNet, COCO, and PASCAL VOC. Overall, ResNet152v2 is a robust and adaptable deep learning architecture that is still heavily utilized in the field of computer vision research.

[13][9][20]

Spectral Normalization

The stabilization and convergence of neural network models during training can be improved using the deep learning approach known as Spectral Normalization. .

Essentially, Spectral Normalization forces each weight matrix's greatest eigenvalue to be a constant number, usually 1. This helps to regularize the model and minimize overfitting. It also helps to prevent the weights from growing too high, which can lead to instability during training.

Convolutional and fully connected layers in a neural network are two examples of layer types that Spectral Normalization can be beneficial for. It may be used in a variety of methods, such as by multiplying the weight matrix by its spectral norm or by estimating the spectral norm using a power iteration approach before rescaling the weight matrix.

Overall, Spectral Normalization is an effective method for enhancing the performance and stability of deep learning models, especially those that include convolutional layers, which are frequently employed in applications for image and video processing.

[20][18]

3.4 Optimizing Model Selection

Data augmentation techniques, such as horizontal flip and random rotation, are used to increase the diversity and size of the training dataset by generating new training samples from existing ones. The main advantage of data augmentation is that it helps to reduce overfitting, which happens when a model does well on the training set but poorly on new, unseen data.

By applying horizontal flip, the model is exposed to images that are flipped versions of the original, which increases the diversity of the dataset and helps the model to generalize better to new images. Similarly, by rotating the images randomly, the model is exposed to different angles of the same object, which also helps to improve its ability to recognize the object from various perspectives.

In addition to horizontal flip and rotation, other data augmentation techniques, such as zoom-range, shear-range, and rescale, were used in this example to further increase the diversity of the training dataset and improve the performance of the model.

Zoom-range allows the model to see the object of interest at different scales by randomly zooming in or out on the images during training. This can help the model to recognize objects that are not in the center of the image or that are partially occluded. Similarly, shear-range applies random affine transformations to the images, which is any transformation that keeps attention to the ratios of distances and collinearity (all points initially located on a line remain there after the transformation). This can help the model to recognize objects that are tilted or slanted in the image.

Finally, rescale=1./255 is used to normalize the pixel values of the images between 0 and 1, which helps to ensure that the model is less sensitive to variations in brightness and contrast across different images.

By combining these data augmentation techniques, the diversity and size of the training dataset are increased, which can help the model to generalize better to new, unseen images and reduce overfitting. The increase in accuracy from around 50 percent to over 80 percent in this example indicates that the data augmentation techniques used were effective in improving the performance of the model.

After running over 35 models with each taking 5-10 hours to run, I found that the best model using sequential library in Keras consisted of 16 layers as shown below (figure 8, 9).

Out[8]:

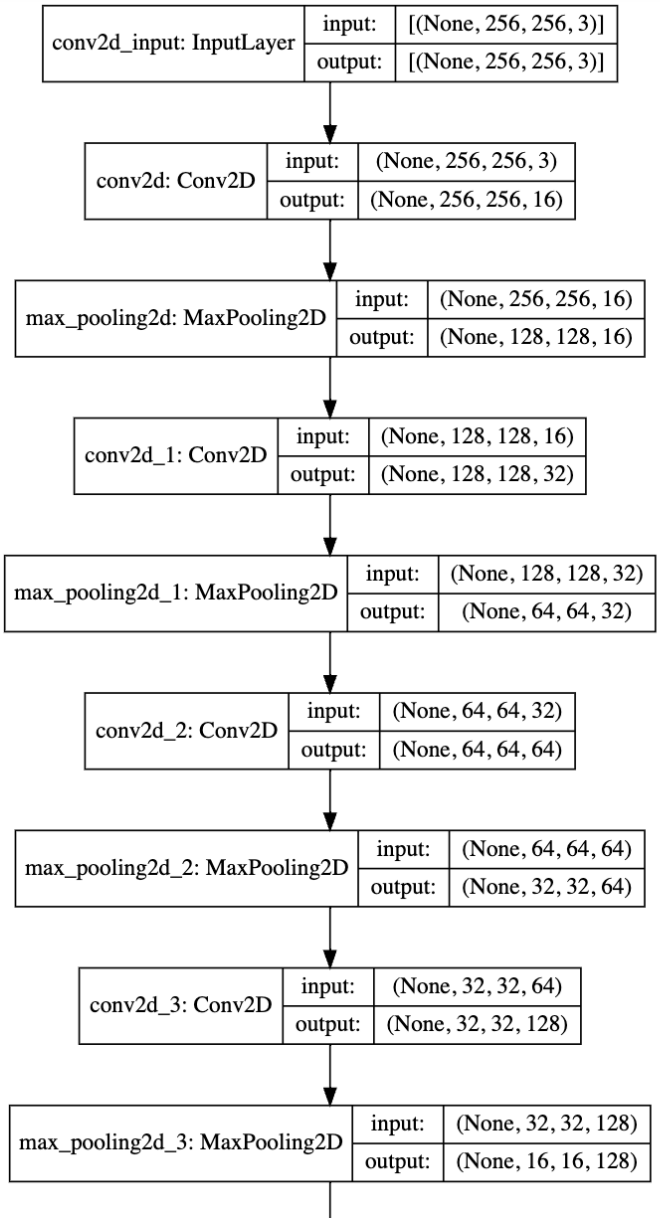

Figure 8: 'Model Architcture Flowchart'

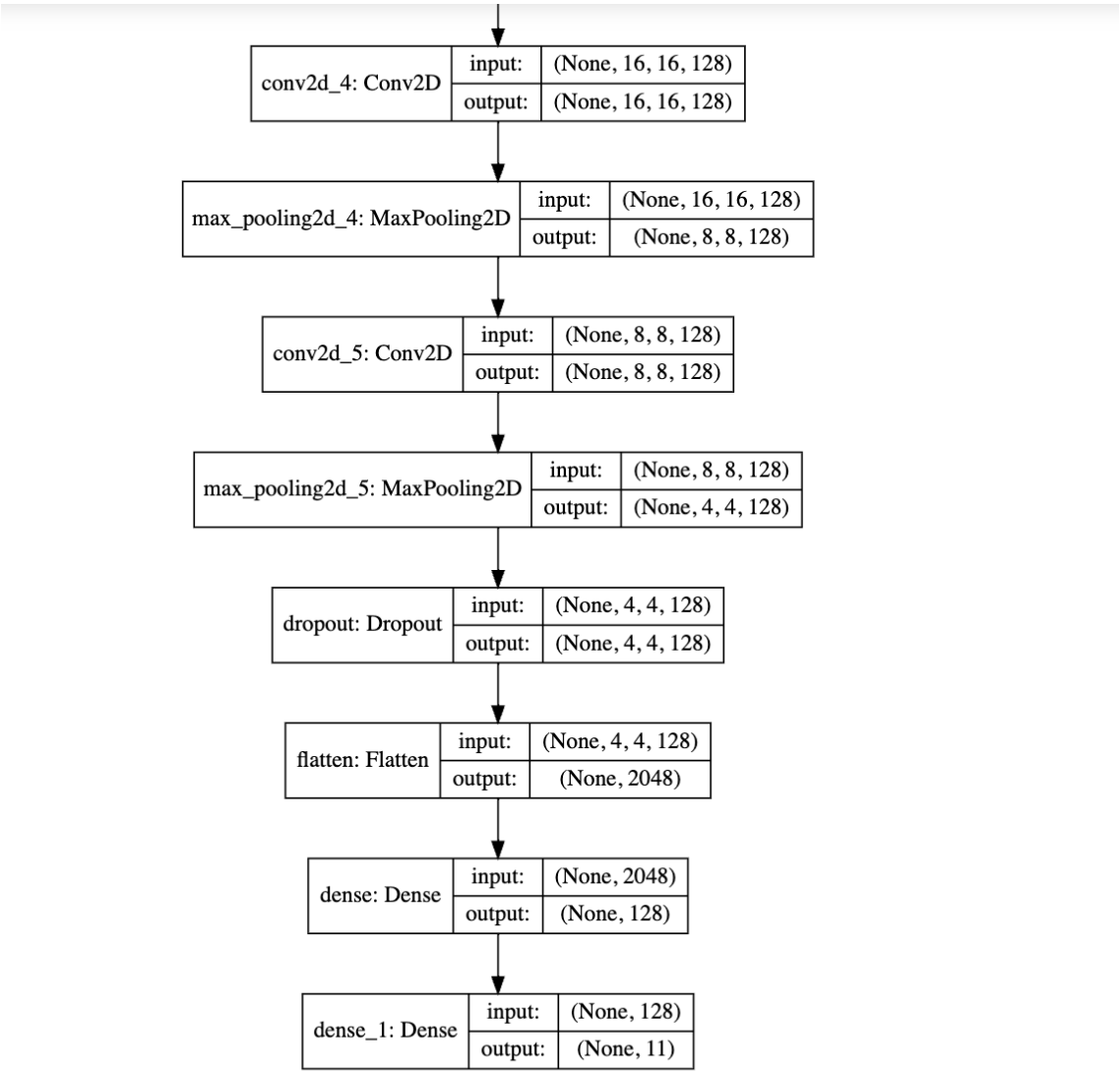

Figure 9: 'Continuation of Model Architcture Flowchart'

The model comprises of a set of six convolutional layers with the ReLU activation function and padding set to 'same', ensuring that the output feature maps have the same spatial dimensions as the input. In deeper levels, the size and quantity of filters steadily increase. Following the convolutional layers are the max-pooling layers, which take the highest value possible inside each pooling window to minimize the spatial dimensions of the feature maps.

The order of the convolution layers and the choice of kernel size can have a major impact on the performance of a convolutional neural network (CNN). In this particular case, it has been observed that using a convolution layer with 128 filters, the 7x7 kernel at the first layer, and the 3x3 kernel at the last layer improves the accuracy and overall performance of the CNN.

The choice of kernel size is an important consideration in CNNs as it determines the size of the receptive field of each neuron in the layer. A larger kernel size, such as the 7x7

kernel used in this case, has a wider receptive field and is better suited for capturing global features in the input image. On the other hand, a smaller kernel size, such as the 3x3 kernel, has a smaller receptive field and is better suited for capturing local features and details.

When the 7x7 kernel is used at the first layer of the CNN, it allows the network to capture larger and more complex features in the input image at an early stage of the network. This, in turn, can help the subsequent layers to focus on more specific and finer details in the input image. On the other hand, using the 3x3 kernel at the last layer can help the network to capture more precise features and patterns, thereby improving the accuracy of the model.

Also, increasing the number of filters beyond 128 in this example led to overfitting, where too much complexity causes the model to remember training data rather than learning broad patterns that may be applied to fresh data. Therefore, it is crucial to balance the number of filters with the complexity of the model and the amount of training data available to achieve optimal performance.

After the convolutional and max-pooling layers, a dropout layer with a rate of 0.4 is added, which randomly drops out 40 percent of the neurons during training to prevent overfitting. The flattened output is passed through two dense layers with ReLU activation function and a final dense layer with a softmax activation function, which outputs the classification probabilities for each of the 11 classes.

Next, the model is compiled using the categorical cross-entropy loss function, which is commonly used for multi-class classification problems. The Adam optimizer is used with a learning rate of 0.001, which adapts the learning rate during training to improve the convergence speed and stability.

During training, the model is monitored using the accuracy metric, which measures the proportion of correctly classified images out of all images. The training is stopped early if there is no improvement in the validation loss for a certain number of epochs, which is set to the default value of 10. The model also uses the EarlyStopping library to restore the weights of the model with the best accuracy.

Additionally, the model uses the Adam optimizer with a learning rate of 0.001, which is a popular choice for deep learning tasks due to its adaptive learning rate and momentum-based approach. The loss function is set to categorical cross-entropy, which is commonly used for multi-class classification problems. The metric used to evaluate the model performance during training is Accuracy.

The model is trained for 60 epochs because after many tweaks and trainings, it was noticed that almost all the models start overfitting after 60 epochs.

Overall, this model architecture and training setup appears to be well-suited for the specific task of detecting weather images, as demonstrated by its high accuracy on the validation set. However, it's important to note that the performance of the model may vary depending on the specific characteristics of the dataset and the particular task at hand. It's always a good practice to perform additional experimentation and analysis to ensure that the model is performing optimally.

4 Results

After trying around 35 different models and ideas to improve accuracy, I found the following:

• Including max-pooling after every convolution operation led to an increase in accuracy and without using it the model would underperform.

• It was concluded that the best the activation function in the convolutional layers was ReLU and other activation functions led to a decrease in performance.

• Thirdly, I found that starting with a high kernel size or ending with a high kernel size also decreased the performance, but when using a 7x7 kernel for example in the middle of the convolutions improved accuracy. When only 7x7 kernels were used, the model got complicated and had extra parameters and under performed as a matter of fact.

• The higher the trainable parameters the more the model can be complex and better. However, in many cases this led to overfitting.

• The best transfer learning model on the data was ResNet152v2 with an 86 percent over validation data.

The had an overall of 1,246,123 parameters and all were trainable parameters, while ResNet152v2 had more than 50 million parameters of which only around 500,000 were trainable parameters. The number of parameters demonstrates further why ResNet is more complex and takes longer time.

It is very important to do data augmentation as it massively improves the performance of the model. The figures below (10, 11, 12 and 13) show a comparison of accuracy and loss for the model before data augmentation and after. The X axis here denotes the number of epochs and the Y axis denotes either the accuracy or the loss according to the graph. Higher Accuracy and lower Loss means better performance.

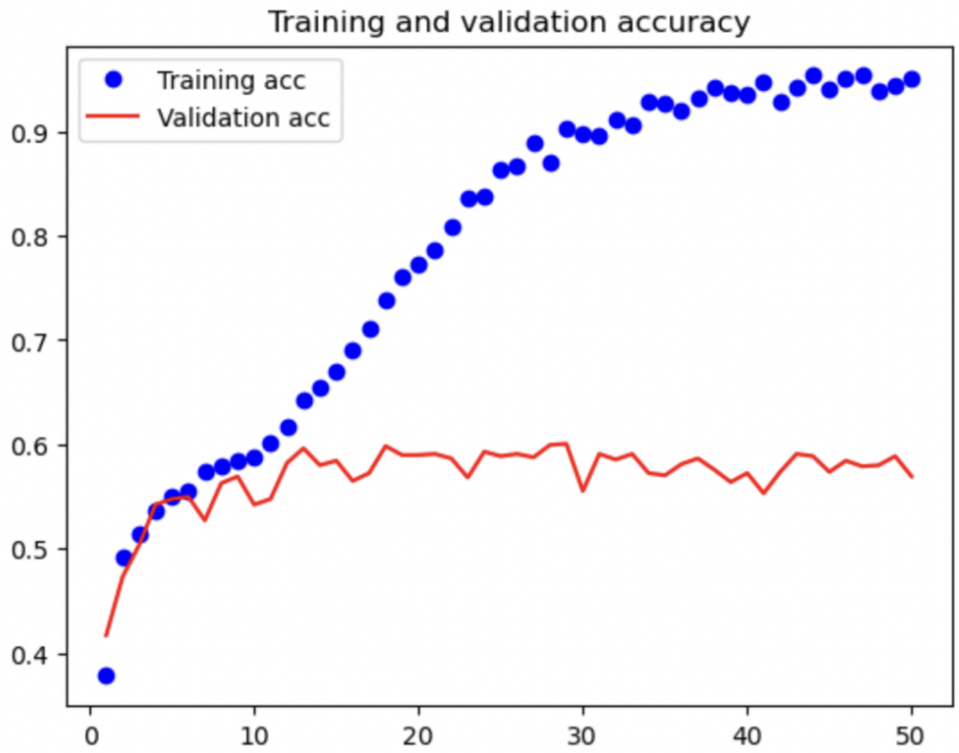

Figure 10: 'Accuracy before data augmentation'

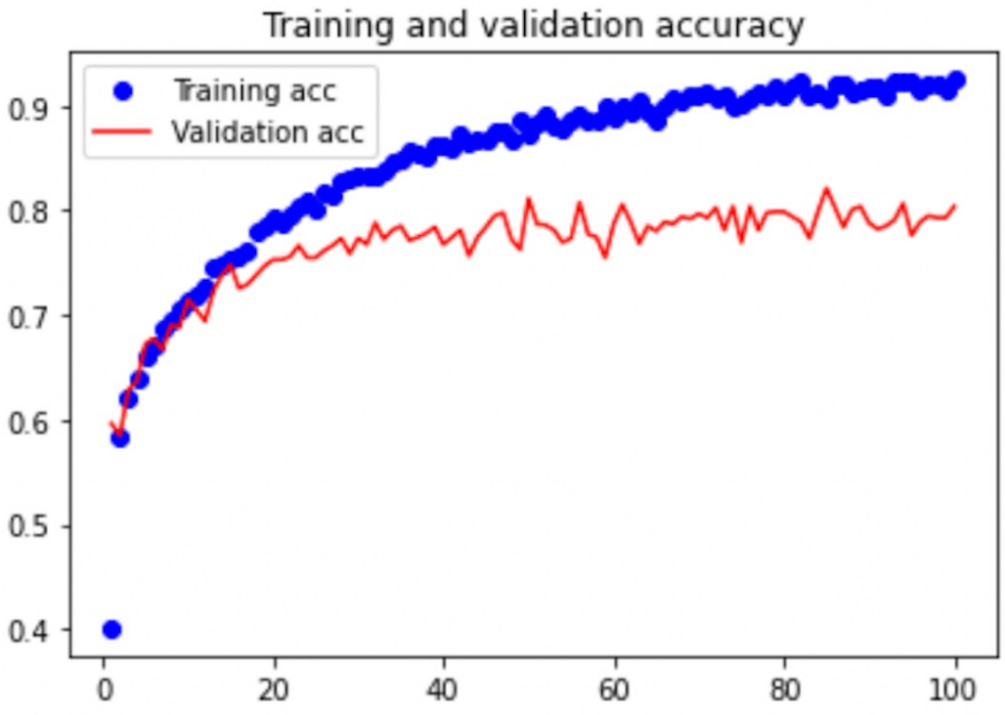

Figure 11: 'Accuracy after data augmentation'

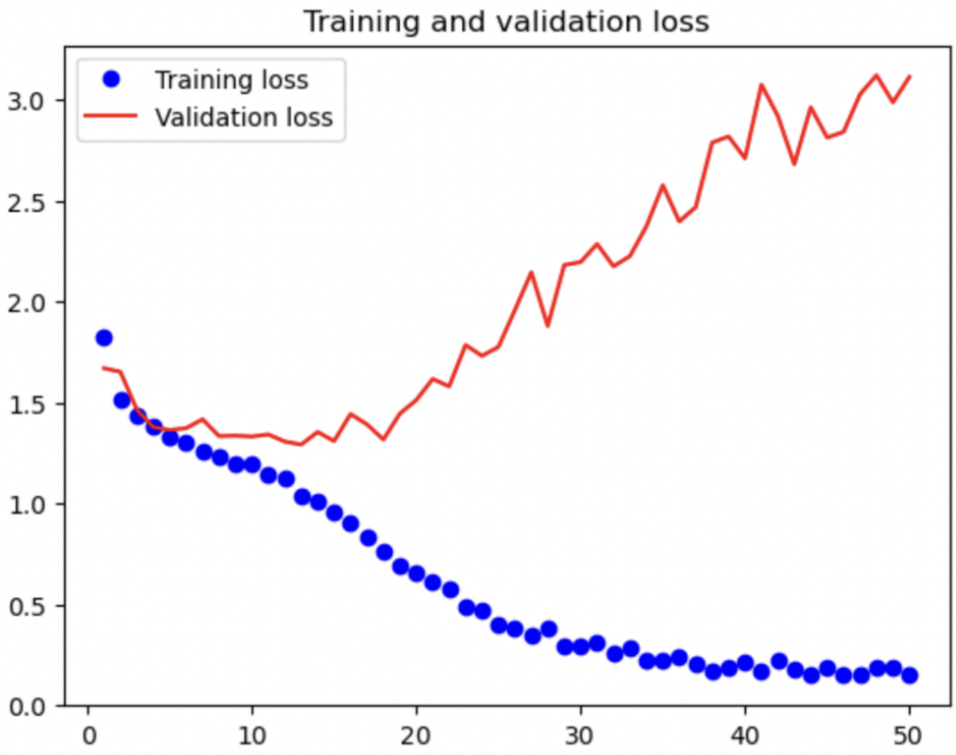

Figure 12: 'Loss before data augmentation'

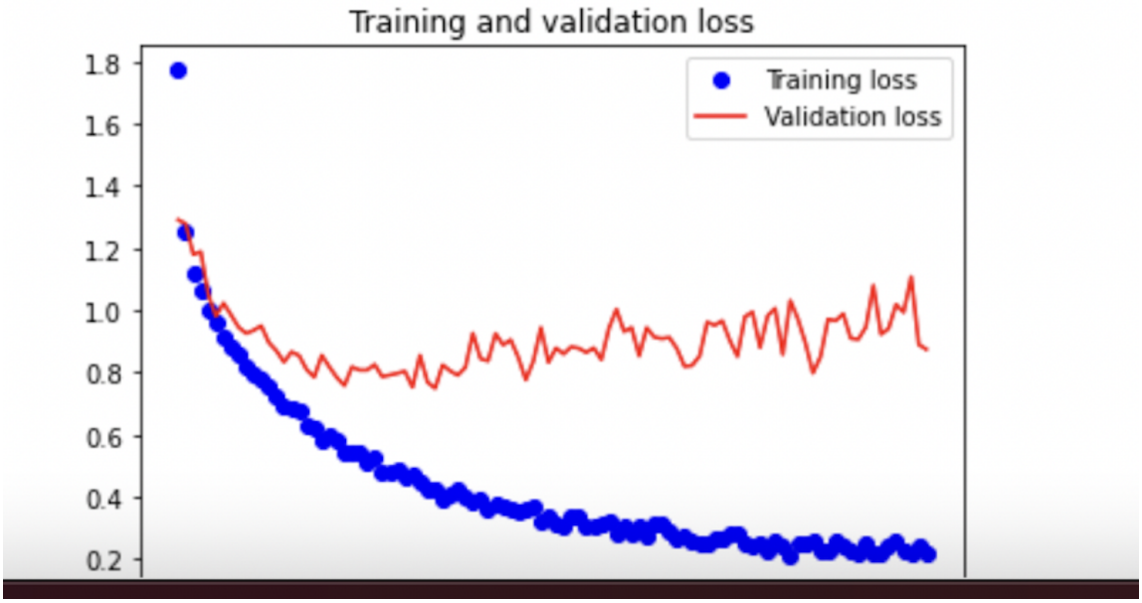

Figure 13: 'Loss after data augmentation'

As it can be seen, the model prior to data augmentation was highly overfitting since the training accuracy keeps improving but the validation does not with a big gap. The training dataset reached a 90 percent accuracy, while the validation accuracy didn't reach 60 percent. After augmentation, the gap between validation accuracy and training accuracy shrunk, which demonstrates an improvment in the model on unseen data. With the validation accuracy reaching 81 percent.

For the loss graphs, you can see that in both cases the training loss was low. However, for validation loss prior to data augmentation was high and started highly increasing after 10 to 15 epochs. This got better after data augmentation with the loss stabilizing for the validation data.

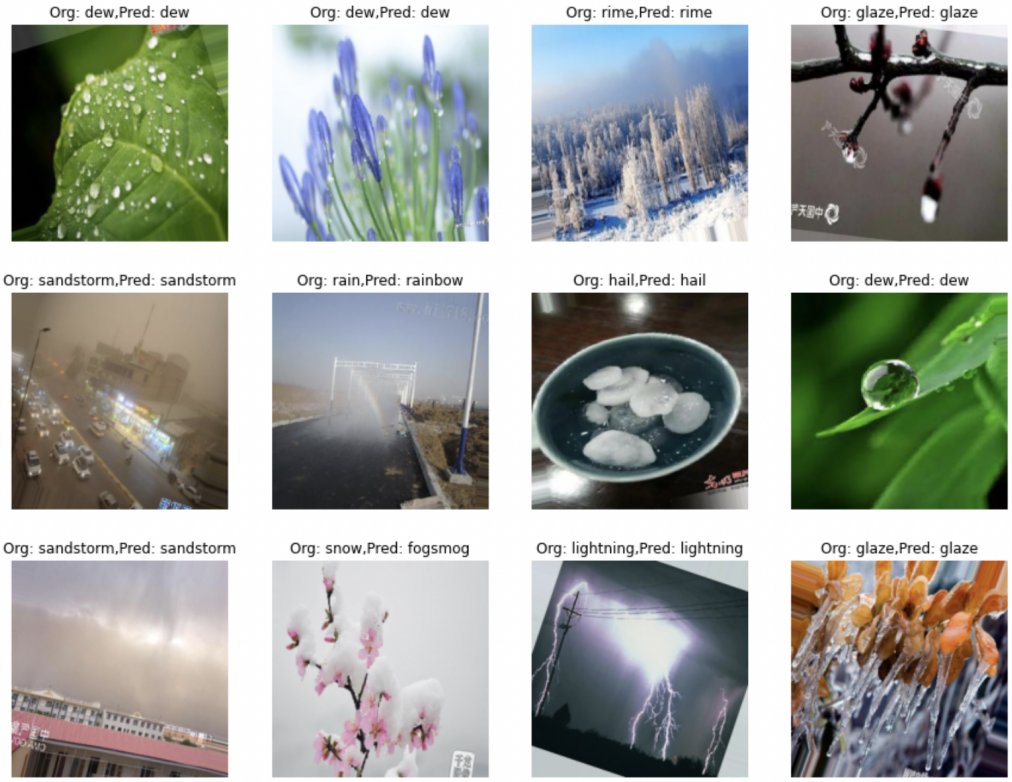

Figure 14: 'Results by the ResNet152v2 model'

[12]

After some analysis of these images, it was found that some of the images with incorrect results can be due to the fact that the image has more label. For example, in figure 14 row 2 image 2 and row 3 image 2 where the model had wrong predictions, it can be seen that there was more than 1 weather phenomena. In row 2 image 2, there's both rain and rainbow and in row 3 image 2, there is a fog in the background. This demonstrates that classifying the images into more than one class might be beneficial for more accurate predictions, which will of course has its own challenges as multi-output classification is much more complex.

These images however were all from the same dataset and the best way to test a model is to test it on other images. That's why I decided to use the model on some real images, some of which taken during this year in Bremen, Germany. I ran the model on these

pictures and the results are shown below.

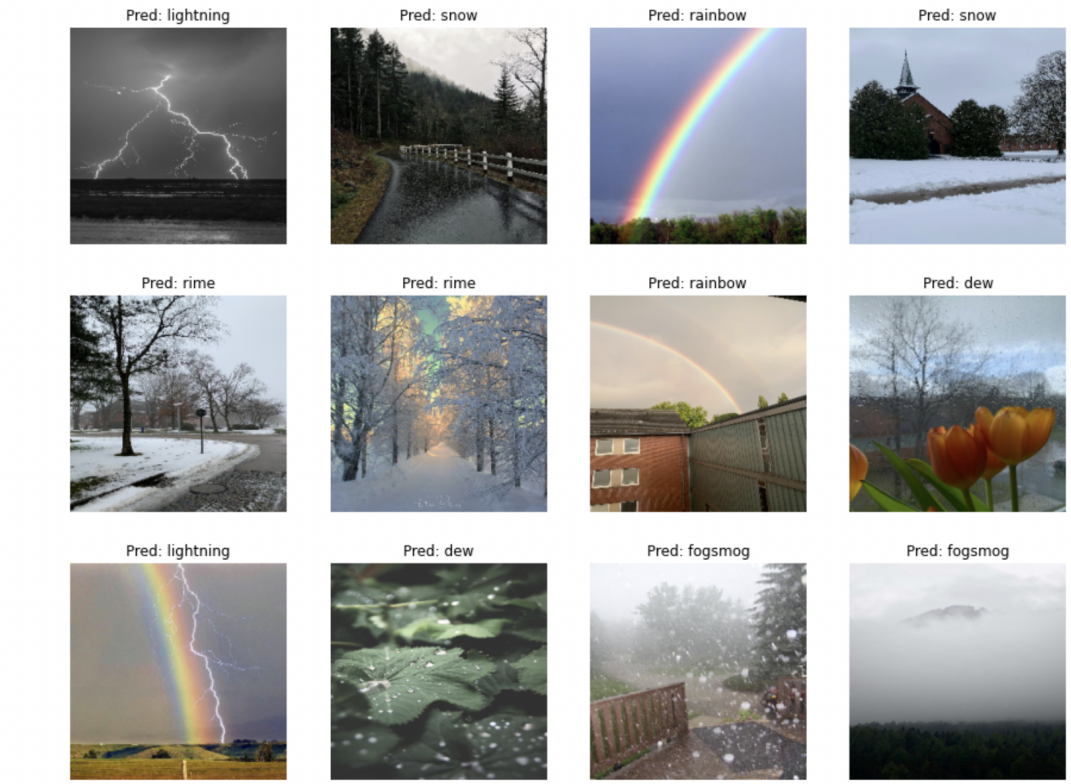

Figure 15: 'Results on out-of-distribution data'

[12]

5 Conclusion

In conclusion, there has been a lot of research done regarding weather classification using image detection as it is a very important and tricky problem to solve. With thousands of simulation data collected everyday, it is still not possible to predict the weather conditions accurately enough. Weather classification helps with this process. The paper discussed deep neural networks used for image detection and a comparison with a model created here for this sample dataset consisting of over 6800 images. The model was able to predict the condition of the weather with over 81 percent accuracy for validation data. When tested with random pictures with a specific weather condition the model was able to identify it without being trained for these specific scenarios which shows the resilience of the model as well as its quality. The transfer model best suited for this data was concluded to be ResNet152v2 with over 86 percent validation accuracy. After visualizing how the model performs on the data, it was shown that one of the main issues with improving on this model was the fact that more than one weather condition or meteorological phenomenon can be happening at the same time and thus creating a model to classify to multiple outputs might be a good solution for improvement.

References

[1] Christian Szegedy et al. “Going deeper with convolutions”. 2015. DOI: DOI:10.1109/CVPR.2015.7298594.

[2] Deepanshi. “Beginners Guide to Artificial Neural Network”. 2021. URL: https://www.analyticsvidhya.com/blog/2021/05/beginners-guide-to-artificial-neural-network/.

[3] Mohamed Elhoseiny, Sheng Huang, and Ahmed Elgammal. “Weather classification with deep convolutional neural networks”. 2015. DOI: DOI:10.1109/ICIP.2015.7351424.

[4] Jiayi Fan, Jang Hyeon Lee, and YongKeun Lee. “Application of Transfer Learning for Image Classification on Dataset with Not Mutually Exclusive Classes”. 2021, pp. 1–4. DOI: 10.1109/ITC-CSCC52171.2021.9501424.

[5] Jose Carlos Villarreal Guerra et al. “Weather Classification: A new multi-class dataset, data augmentation approach and comprehensive evaluations of Convolutional Neural Networks”. 2018. DOI: 10.1109/AHS.2018.8541482.

[6] Kaiming He et al. “Deep Residual Learning for Image Recognition”. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[7] C. Sun et al J. Huang V. Rathod. “Speed/accuracy trade-offs for modern convolutional object detectors”. 2016. DOI: arXiv:1611.10012,pp.1-21,2016.

[8] N. Japkowicz and S. Stephen. “The class imbalance problem: A systematic study”. 2002. DOI: https://doi.org/10.3233/IDA-2002-6504.

[9] S. Komal Kaur, T. Adilakshmi, and T. Jalaja. “A Residual Network Model (ResNet152) for Bronchitis Detection using X-ray Images”. 2022, pp. 22–27. DOI: 10.1109/ICPS55917.2022.00012.

[10] Cewu Lu et al. “Two-Class Weather Classification”. 2014, pp. 3718–3725. DOI: 10.1109/CVPR.2014.475.

[11] Piyush Nagpal, Shivani Atul Bhinge, and Ajitkumar Shitole. “A Comparative Analysis of ResNet Architectures”. 2022, pp. 1–8. DOI: 10.1109/SMARTGENCON56628.2022.10083966.

[12] Keiron O’Shea and Ryan Nash. “An Introduction to Convolutional Neural Networks”. 2015. DOI: https://doi.org/10.48550/arXiv.1511.08458.

[13] Xiaosong Zhang Riaz Ullah Khan and Rajesh Kumar. “Analysis of ResNet and GoogleNet models for malware detection”. 2019. DOI: 10.1109/SPIN.2018.8474198.

[14] Sumit Saha. “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way”. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional%20neural-networks-the-eli5-way-3bd2b1164a53.

[15] M. G. Schultz et al. “Can deep learning beat numerical weather prediction?” 2021. DOI: https://doi.org/10.1098/rsta.2020.0097.

[16] R. Shanmugamani. “Deep Learning for Computer Vision.” 2018.

[17] Brandon Da Silva. “Playing Super Mario Bros with Proximal Policy Optimization”. 2018. URL: https://brandinho.github.io/mario-ppo/.

[18] et al. Takeru Miyato. “Spectral Normalization for Generative Adversarial Networks”. DOI: https://doi.org/10.48550/arXiv.1802.05957.

[19] J. Talukdar et al. “Transfer Learning for Object Detection using State-of-the-Art Deep Neural Networks”. 2018, pp. 78–83. DOI: 10.1109/SPIN.2018.8474198.

[20] “Tensorflow, Keras Documentation”. URL: https://www.tensorflow.org/guide/keras.

[21] Haixia Xiao. “Classification of Weather Phenomenon From Images by Using Deep Convolutional Neural Network”. 2021. DOI: 10.1029/2020EA001604.

[22] Xue Ying. “An Overview of Overfitting and its Solutions”. 2019. DOI: DOI:10.1088/1742-6596/1168/2/022022.