Weather Classification

INTRODUCTION:

Weather significantly impacts daily life, affecting everything from crop destruction to commute disruptions. Traditionally, weather detection relies on expensive sensors and human involvement, often resulting in low accuracy. While humans can easily determine the weather from a single image, designing an automated classifier to do this is challenging. To address constraints on human resources and expensive instruments, a new approach using convolutional neural networks (CNNs) has been proposed. This method leverages readily available surveillance cameras to capture outdoor images and classify weather conditions based on those images. By analysing photometric properties such as colour and texture, the proposed CNN architecture can accurately detect weather conditions from an image.

PROBLEM STATEMENT:

To determine the weather condition by utilizing Convolutional Neural Networks. Classify it into cloudy, sunny, rainy, shine or sunrise.

DATASET:

The dataset features 5 different classes of weather collected from kaggle, however it's real life data so any system for weather classification must be able to handle this sort of images. Training set includes about 1500 labelled images including the validation images. Images are not of fixed dimensions and the photos are of different sizes. Images do not contain any border.



Data augmentation:

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model.



The above figure is an example of an augmented image.

**Convolutional Neural Network (CNN)**

A convolutional neural network (CNN or ConvNet) is a type of deep neural network commonly used for visual imagery analysis. Unlike traditional fully connected networks (multilayer perceptrons), CNNs use a mathematical operation called convolution instead of general matrix multiplication in at least one of their layers.

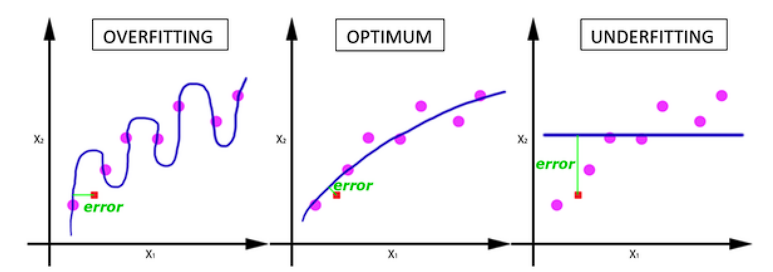
**CNN Architecture**

A CNN has an input layer, an output layer, and multiple hidden layers. The hidden layers include various specialized types:

1. **Convolutional Layer**: Detects edges, lines, colors, and other visual elements by applying filters to the input image. Key parameters are the number of filters, filter size, stride, padding, and activation functions (e.g., ReLU).
2. **MaxPooling Layer**: Reduces the dimensionality of the image by replacing an area (nxn pixels) with the maximum pixel value from that area, thus downsampling the image.
3. **Dropout Layer**: Prevents overfitting by randomly setting a fraction of input units to 0 during training, forcing the network to learn more robust features.
4. **Flatten Layer**: Converts the 2D matrix output from the convolutional layers into a 1D vector, preparing it for the Dense layer.
5. **Dense Layer**: Fully connected layers that map the features extracted by convolutional layers to the final output.
6. **Batch Normalization Layer**: Normalizes the inputs of each layer to have a unit Gaussian distribution. This helps in initializing the network properly, makes training faster and more stable, and reduces overfitting.

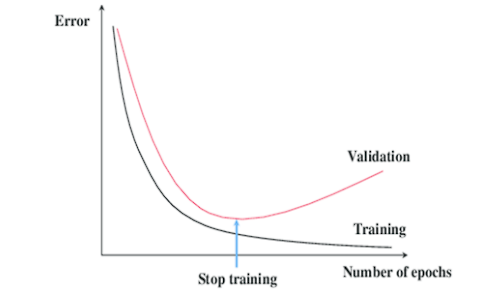
**Epochs:**

In terms of artificial neural networks, an epoch refers to one cycle through the full training dataset. Usually, training a neural network takes more than a few epochs. In other words, if we feed a neural network the training data for more than one epoch in different patterns, we hope for a better generalization when given a new data. With a neural network, the goal of the model is generally to classify or generate material which is right or wrong. Thus, an epoch for an experimental agent performing many actions for a single task may vary from an epoch for an agent trying to perform a single action for many tasks of the same nature.



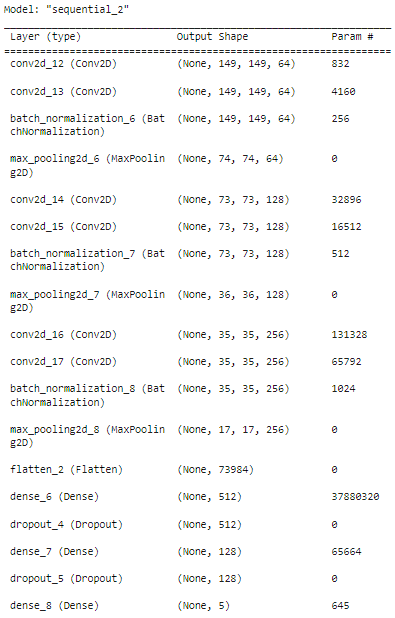
**Early Stopping:**

One of the simplest methods to prevent overfitting of a network is to simply stop the training before overfitting has had a chance to occur. It comes with the disadvantage that the learning process is halted.Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. Early stopping is a method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a hold out validation dataset.

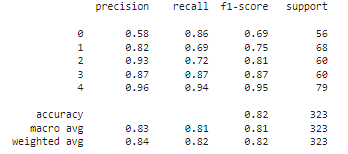


**FINAL MODEL:**

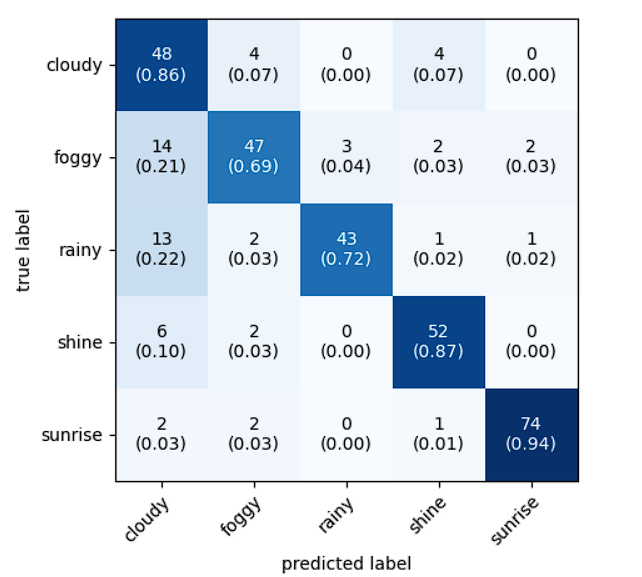
The final model consists of 3 blocks of 2 convolutional layers followed by batch normalization and max pooling layer. After that there is a flatten to introduce a full connection between the units, dropout layers to reduce overfitting and dense layers that slowly decrease the size of the array from 512 to 5.



An early stopping callback was also used in case overfitting happened. The growth of model accuracy and loss during training and validation is given below. The confusion matrix of model performance is also included below.

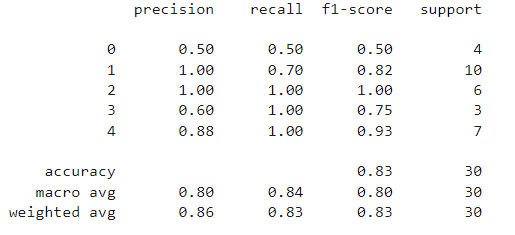


Classification Report

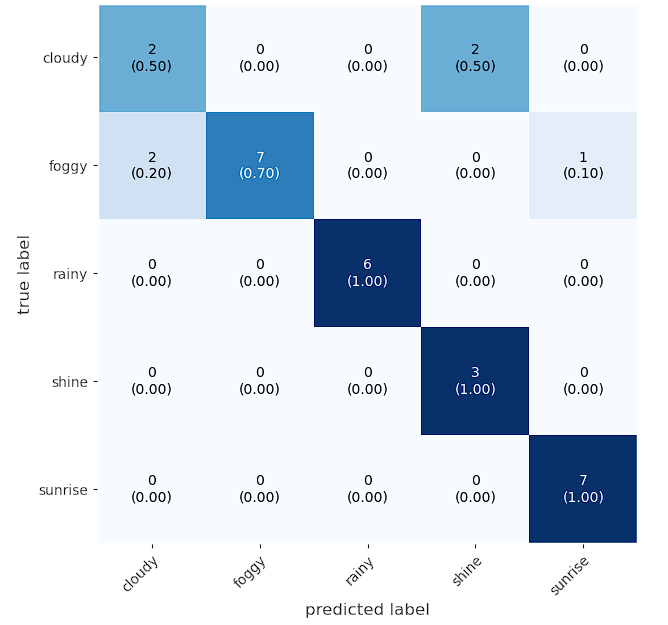
  
 Confusion Matrix

Testing:

Testing data includes 30 images of all the 5 different classes.



Classification Report



Confusion Matrix

**CONCLUSION:**

Our objective was to develop a CNN model for weather detection with tensorflow. After the project we were able to complete the objective to develop a powerful for classifying images as rainy, cloudy, shiny, foggy and sunrise.

**REFERENCES:**  
<https://www.tensorflow.org/>

<https://scikit-learn.org/>

<https://numpy.org/>