Real-Time Weather Detection and Classification with Convolutional Neural Networks

Miguel Marines - A01705317

Problem

Real-time weather information is particularly important for those who work outdoors, such as construction workers, farmers, or emergency responders. These individuals need to be aware of any sudden weather changes that could put them at risk. For example, if a severe storm is approaching, construction workers need to halt their work and seek shelter, while farmers need to protect their crops. Real-time weather updates can also help commuters plan their routes and avoid dangerous road conditions during inclement weather.

In addition, real-time weather information is vital for businesses that are directly impacted by weather conditions, such as airlines, shipping companies, and outdoor event organizers. Knowing the current weather conditions and forecasts can help these businesses make informed decisions about operations, scheduling, and logistics. For instance, airlines need to adjust their flight schedules to avoid turbulence or severe weather conditions, while outdoor event organizers may need to postpone or cancel events due to safety concerns. Therefore, real-time weather information is essential for businesses to minimize the impact of weather-related disruptions and maintain their operations.

Project

This deep learning project aims to identify weather patterns using convolutional neural networks (CNN). This project involves training a CNN on a dataset of weather images to recognize various weather conditions such as sunny, cloudy, rainy, or snowy. Once the CNN is trained, it can be used to identify weather conditions in real-time using live images captured by weather cameras and other sensors. This will allow for more accurate and timely weather forecasting and to predict and respond to weather-related events.

Dataset

This project uses a dataset containing sky images of five weather classes: sunny, cloudy, rain, snow, and electrical storm. Each class contains 140 images for training and 60 images for testing. The dataset is used to train a machine learning model, involving convolutional neural networks, to accurately identify the different weather conditions in images.

The images are in format RGB with a size of 300 x 300 pixels and a resolution of 144 pixels / inch.

Sunny Example:

Cloudy Example:

Rain Example:

Snow Example:

Electrical Storm Example:

Convolutional Neural Network Model

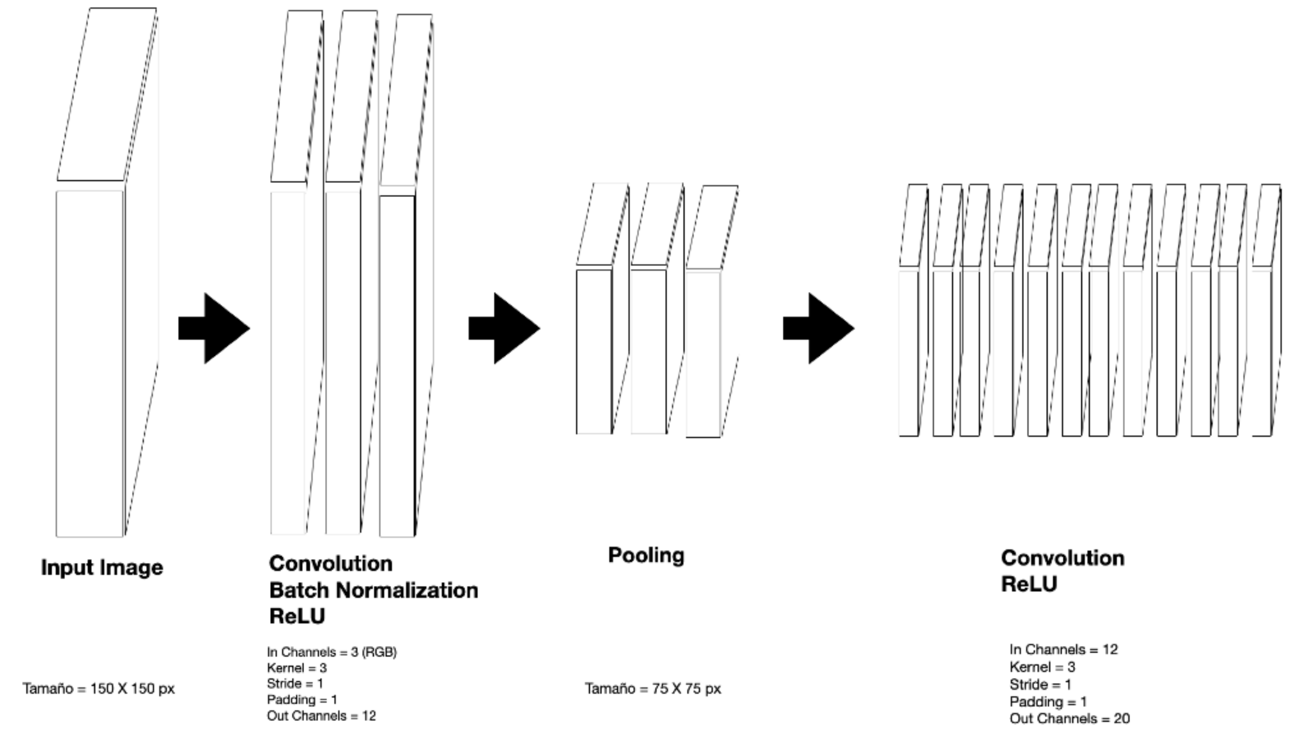
This project implements a deep learning model that is composed of a convolutional neural network. The architecture starts with a convolution with three in-channels (RGB), a three size kernel, a stride of 1, a padding of 1 and twelve out-channels. Additionally, batch normalization and the activation function ReLU are applied.

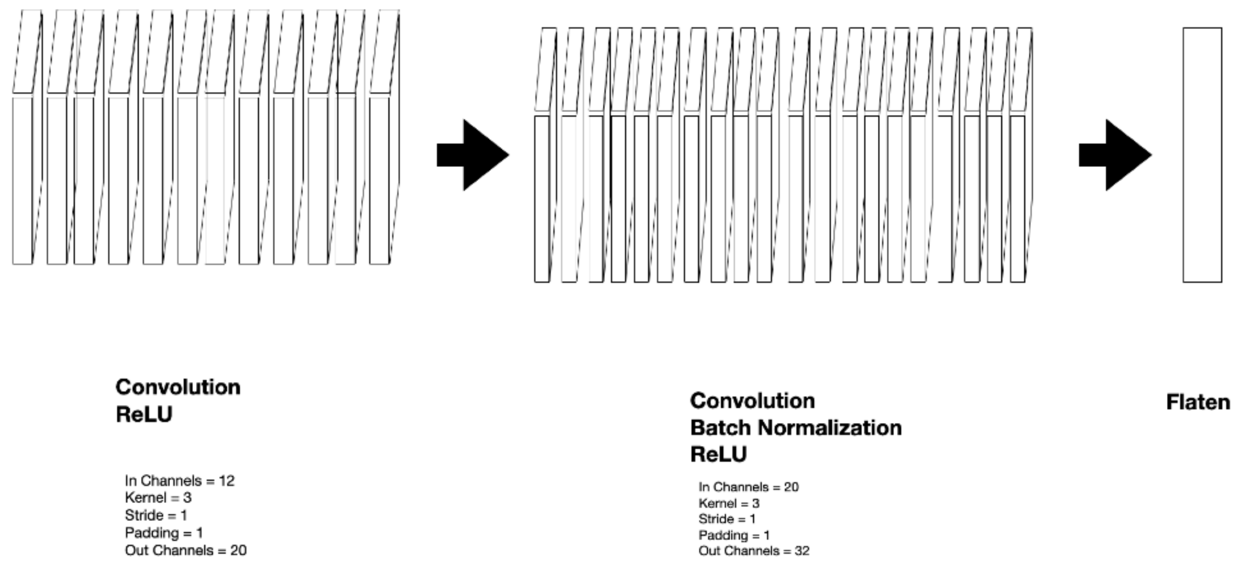
After the first convolution, a padding of value two is applied two reduce the initial images size in half. Furthermore, a second c convolution is applied with twelve in-channels, a three size kernel, a stride of 1, a padding of 1 and twenty out-channels. Additionally, the activation function ReLU is applied.

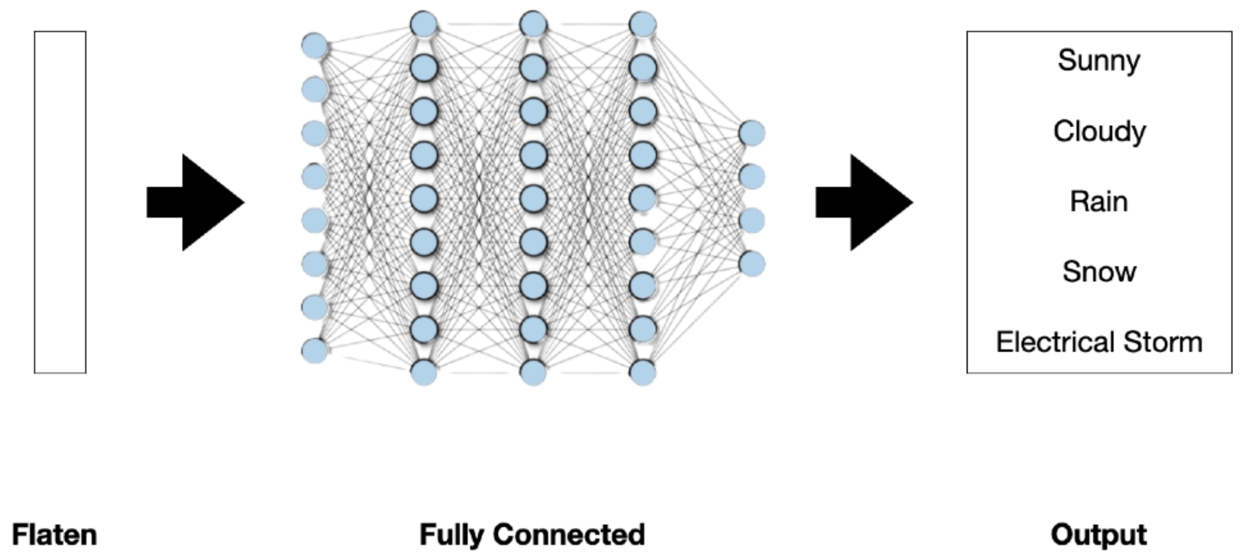
A third convolution is applied in this model with with twenty in-channels, a three size kernel, a stride of 1, a padding of 1 and thirty-two out-channels. Additionally, batch normalization and the activation function ReLU are applied.

Subsequently, the flatten is applied. Then, the fully connected layer takes in the flattened output and applies a linear transformation to it. This linear transformation consists of a matrix multiplication between the input and a weight matrix, followed by an addition of a bias vector. The resulting output is then passed through a non-linear activation function such as ReLU to introduce non-linearity into the model.

Finally, the output is obtained with the final weights to be able to make predictions of images and catalogue them according to the different classes.


Input Image
Tamaño = 150 X 150 px
Convolution
Batch Normalization
ReLU
In Channels = 3 (RGB)
Kernel = 3
Stride = 1
Padding = 1
Out Channels = 12
Pooling
Tamaño = 75 X 75 px
Convolution
ReLU
In Channels = 12
Kernel = 3
Stride = 1
Padding = 1
Out Channels = 20


Convolution
ReLU
In Channels = 12
Kernel = 3
Stride = 1
Padding = 1
Out Channels = 20
Convolution
Batch Normalization
ReLU
In Channels = 20
Kernel = 3
Stride = 1
Padding = 1
Out Channels = 32
Flaten



Training and Testing

Batch Size Training = 20

Batch Size Testing = 8

Epochs = 30

Learning Rate = 0.001

Weight Decay = 0.0001

Implementation

To develop this project the programming language python was used due to its simplicity and versatility. Additionally, python was selected for this project because of its easy-to-read syntax and large number of libraries, including powerful machine learning frameworks such as TensorFlow, PyTorch, and scikit-learn.

For the development of the deep learning model PyTorch was used because it has a growing popularity in the scientific community and the majority of the newest deep learning models are being developed with this library.

Results

<u>Training and Testing</u>

```
Epoch: 0     Train Accuracy: 0.6275071633237822     Test Accuracy: 0.5095890410958904
Epoch: 1     Train Accuracy: 0.8051575931232091     Test Accuracy: 0.7945205479452054
Epoch: 2     Train Accuracy: 0.8925501432664756     Test Accuracy: 0.8082191780821918
Epoch: 3     Train Accuracy: 0.8853868194842407     Test Accuracy: 0.7780821917808219
Epoch: 4     Train Accuracy: 0.9169054441260746     Test Accuracy: 0.863013698630137
Epoch: 5     Train Accuracy: 0.9555873925501432     Test Accuracy: 0.6958904109589041
Epoch: 6     Train Accuracy: 0.9441260744985673     Test Accuracy: 0.8410958904109589
Epoch: 7     Train Accuracy: 0.9598853868194842     Test Accuracy: 0.7917808219178082
Epoch: 8     Train Accuracy: 0.9555873925501432     Test Accuracy: 0.8986301369863013
Epoch: 9     Train Accuracy: 0.9828080229226361     Test Accuracy: 0.852054794520548
Epoch: 10    Train Accuracy: 0.9856733524355301     Test Accuracy: 0.873972602739726
Epoch: 11    Train Accuracy: 0.9799426934097422     Test Accuracy: 0.8273972602739726
Epoch: 12    Train Accuracy: 0.9627507163323782     Test Accuracy: 0.873972602739726
Epoch: 13    Train Accuracy: 0.997134670487106      Test Accuracy: 0.8849315068493151
Epoch: 14    Train Accuracy: 0.995702005730659      Test Accuracy: 0.8547945205479452
Epoch: 15    Train Accuracy: 0.997134670487106      Test Accuracy: 0.8794520547945206
Epoch: 16    Train Accuracy: 0.9885386819484241     Test Accuracy: 0.8246575342465754
Epoch: 17    Train Accuracy: 0.9813753581661891     Test Accuracy: 0.8794520547945206
Epoch: 18    Train Accuracy: 0.9541547277936963     Test Accuracy: 0.8438356164383561
Epoch: 19    Train Accuracy: 0.9699140401146131     Test Accuracy: 0.873972602739726
Epoch: 20    Train Accuracy: 0.9684813753581661     Test Accuracy: 0.810958904109589
Epoch: 21    Train Accuracy: 0.9570200573065902     Test Accuracy: 0.8082191780821918
Epoch: 22    Train Accuracy: 0.9699140401146131     Test Accuracy: 0.8246575342465754
Epoch: 23    Train Accuracy: 0.995702005730659      Test Accuracy: 0.8301369863013699
Epoch: 24    Train Accuracy: 0.9842406876790831     Test Accuracy: 0.821917808219178
```

<u>Predictions</u>

```
Snow.jpg -> ['Snow']
Cloudy.jpg -> ['Cloudy']
Sunny.jpg -> ['Sunny']
Electrical Storm.jpg -> [‘Electrical Storm']
Rain.jpg -> ['Rain']
```

References

- Horace He, "The State of Machine Learning Frameworks in 2019", The Gradient, 2019.
- https://www.kaggle.com/datasets/jehanbhathena/weather-dataset

- https://pytorch.org/
- https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/
- https://docs.google.com/document/d/1xwow47lk0-djwzG7EIA27HfIhLUb4BTG/edit
- https://www.geeksforgeeks.org/introduction-convolution-neural-network/