Weather Conditions Recognition using CNN and Image Classification.

Venkata Sai Likhita Kurapati
Computer Science Engineering
University of Texas at Arlington
Arlington, Texas
vxk6295@mavs.uta.edu

Mayukha Thumiki
Computer Science Engineering
University of Texas Arlington
Arlington, Texas
mxt5616@mavs.uta.edu

***Abstract*—Our paper basically develops a system for weather conditions recognition which is accurate and robust using the CNN and image classification techniques. This recognizes different weather conditions from images, monitoring the environment and predicting the weather for real time weather images and also works well for sparse training data.**

Keywords— CNN, Image Classification, Sparse Training data, Anonymous Test Data

I. INTRODUCTION

Weather conditions recognition is an important component of modern living impacting various domains ranging from transportation and agriculture to urban planning and disaster management. Identifying weather conditions accurately from visual data, such as photographs, has the potential to improve decision-making processes and the safety and efficiency of a wide range of applications.

Deep learning techniques, particularly Convolutional Neural Networks, have revolutionized image identification in recent years, allowing the development of complex models capable of classifying and categorizing visual data. By maximizing their ability to learn discriminative features, handle spatial variations which often deals with requiring models to identify patterns that are invariant to spatial transformations such as rotations, translations, and scaling. It also deals with managing large-scale data. The availability of extensive weather image datasets empowers CNN to learn intricate patterns and inherent to various weather conditions, It can generalize well to unseen images and improve model's adaptability to new and changing weather scenarios. Moreover, it also promotes transfer learning where pre-trained models are fine-tuned for specific tasks, can be repurposed for weather recognition with minor modifications. Furthermore, it potentially integrates multi-model information. CNN and image classification approaches provide a powerful foundation for weather condition recognition. The convergence of these cutting-edge technologies has the potential to improve our understanding of weather events, improve decision-making processes, and contribute to the creation of many innovative applications that are resistant to a wide range of weather conditions.

Dataset: The dataset that we've taken here is the 'Weather Image Recognition' from Kaggle to do our study. This dataset consists of 6862 images of different weather situations, to categorize the weather based on the image. The eleven categories into which the images are divided include dew, fog/smog, frost, glazing, hail, lightning, rain, rainbow, rime, sandstorm, and snow. The images are first cropped to 224 x 224 pixels (IMAGE_SIZE) and put into batches of 32 pictures each (BATCH_SIZE).

The novelty behind our approach in this report is the model we created works decently on the sparse data and can be used to classify when given new test data.

This report aims to delve deeper into the mechanisms and methodologies, showing the results and how CNN and image classification play a key role in emphasizing their functionalities to construct a robust and reliable weather recognition model.

II. RELATED WORK

One of the early studies in subject of weather recognition suggests a CNN -RNN based method that approaches it as a multi-label classification challenge, allowing for the assignment of numerous weather conditions to a single image. The method achieves higher performance in weather recognition by combining a CNN channel-wise attention model with a Recurrent Neural Network (RNN) to capture dependencies across weather classes. Another study, CMWRN outperforms state-of-the-art approaches on benchmark datasets by incorporating multi-scale weather-cue features for precise weather-cue segmentation and introducing a generalized pooling strategy for more discriminative weather representations. Some other works that are related can be:

1. *"WeatherNet: Fine-Grained Weather Classification Using Convolutional Neural Networks" by Li et al. (2018):* The researchers present WeatherNet, a deep learning framework for classifying fine-grained weather conditions in photographs. The research focuses on employing CNN architectures to classify various types of clouds, haze, fog, and rain.
2. *"DeepWeather: A Convolutional Neural Network Approach for Weather Recognition" by Zhu et al. (2020):* This model is proposed by the authors for accurate weather recognition using CNNs and a large-scale dataset. The research focuses on enhancing recognition accuracy in difficult situations such as mist, smog, and gloomy skies.
3. *"Weather Condition Classification Using Transfer Learning and Data Augmentation" by Tariq et al. (2021):* The impact of transfer learning and data augmentation strategies on weather condition categorization using CNNs is investigated in this work. The authors explore different pre-trained CNN architectures and augmentation methodologies for improving recognition performance.
4. "Weather Classification Using Deep Convolutional Neural Networks" by Abadi et al. (2017): This research proposes a CNN-based approach for weather classification based on a dataset of outdoor

photographs recorded under various weather conditions. The authors show how CNNs can tell the difference between bright, cloudy, wet, and snowy circumstances.

III. PROPOSED METHOD

A. CNN architecture:

The ResNet152V2 pre-trained model serves as the foundation of the suggested CNN architecture. This model is skilled at capturing complex features in images using residual blocks for effective deep network handling because it has been pre-trained on a large dataset. The top layers are eliminated, and the weights of the basic model are frozen to prevent overwriting the important learned features. A Sequential model is then built, providing the framework for adding more layers on top of the base.

Three Conv2D layers are added after the base model, each with 128, 64, and 32 filters, respectively. A LeakyReLU activation function with an alpha of 0.1 is applied after each Conv2D layer to add non-linearity and battle the vanishing gradient problem, improving training performance.

Following each Conv2D layer, MaxPooling2D layers are then used. These layers significantly reduce the computational load while keeping critical information pertinent to the classification task by downsampling the feature maps with a pool size of (2, 2). By averaging all values in the feature maps and combining them into a single vector, the next layer, GlobalAveragePooling2D, further reduces spatial dimensions. This process considerably reduces the model's complexity while aiding in the retention of crucial information for classification. Four dense layers with 512, 256, 128, and 64 neurons each are added to the classification head. An activation function follows each Dense layer; the first and fourth layers use ReLU, while the second and third layers use tanh. To reduce overfitting, dropout layers with dropout rates of 0.4, 0.3, 0.2, and 0.2 are consecutively added after the Dense layers. Dropout causes neurons to be randomly deactivated during training, which encourages the network to acquire more universal representations.

The output layer is made up of a Dense layer with n_classes neurons, or the number of classes in the dataset. The output of this layer is transformed into a probability distribution over the classes by the activation function, "Softmax," for this layer. As a result, the model can successfully estimate class probabilities for multi-class classification tasks. Overall, this CNN architecture uses regularization methods, specialized layers, and transfer learning to produce reliable predictions for the target classification task.

B. Multi Class Classification:

With this CNN architecture, a unique classification head is combined with transfer learning. Relevant characteristics are captured by Conv2D layers and pooling procedures, whereas dropout layers and dense layers with activation functions, respectively, introduce non-linearity and regularization. A probability distribution over the classes serves as the model's final output, facilitating multi-class categorization.

The classification procedure entails a number of procedures to map input images to their matching class labels. The initial phase is called feature extraction. Using its previously completed tasks-learned representations, the pre-trained ResNet152V2 base model accepts the input image and extracts high-level features. This method records a variety of intricate patterns and visual clues that are helpful for identifying diverse items and scenes.

C. Convolution and Activation:

Following passage through the basic model, the three Conv2D layers with 128, 64, and 32 filters each further process the feature maps. As they scan the feature maps, the convolutional filters look for distinct patterns in various areas of the images. By allowing certain neurons to have a slight negative slope, the LeakyReLU activation function promotes non-linearity and enables the model to acquire more complex and expressive representations.

The MaxPooling2D layer is used after each Conv2D layer to reduce the spatial dimensions of the feature maps. MaxPooling effectively downsamples the feature maps while maintaining the most important data by choosing the maximum value within a specified pooling zone (2x2 in this case). By reducing the computational load, this technique enables the model to concentrate on key properties.

D. Global Average Pooling:

A GlobalAveragePooling2D layer condenses the spatial dimensions of the feature maps into a single vector after the third Conv2D layer. The most significant aspects in each feature map are effectively summed up by this method, which computes the average of all values in each feature map. The number of parameters is greatly lowered by using global average pooling, which also improves the model's capacity to extrapolate to new data.

The output of the GlobalAveragePooling2D is then fed into four *Dense layers*, together with dropout. ReLU activation function is used by 512 neurons in the first dense layer, which introduces non-linearity and enables the model to learn intricate correlations between features. After the initial Dense layer, a Dropout layer with a rate of 0.4 is added to prevent overfitting. Dropout forces the model to rely on various feature combinations and hence increases its robustness by randomly deactivating 40% of the neurons during training. Additional Dense Layers and Activation are 256, 128 and 64 neurons in the next Dense layers, correspondingly. The tanh activation function is used by the second and third dense layers whereas ReLU is used by the fourth dense layer. To provide more regularization, a Dropout layer with dropout rates of 0.3, 0.2, and 0.2 is placed after each Dense layer.

The number of neurons in the output layer of this CNN architecture is equal to the number of classes in the dataset, which in this instance is 11, which is 11. In the multi-class classification task, each neuron in the output layer corresponds to one of the 11 classes.

IV. EXPERIMENTS

Training: To update the model's parameters (weights and biases), the model is exposed to the training data in batches during the training phase (mini-batch training). Input photos and the matching integer class labels make up the training data. Back-propagation and gradient descent methods are used to iteratively update the model's parameters with the goal of minimizing the specified loss function. A predetermined number of epochs make up the training process; in this case, there are 12 epochs in the training phase.

Loss Function and Optimizer: The model utilizes the 'sparse_categorical_crossentropy' loss function, which is appropriate for multi-class classification tasks using integer class labels. Adam, a well-liked and effective optimization algorithm with a learning rate of 0.001, is the optimizer employed.

A. Hyperparameter Tuning:

The following are the hyperparameters that were used in the model:

1. Learning Rate: The learning rate is set at 0.001 (learning_rate). It controls the size of the weight update steps during optimization, which has an impact on how quickly the model picks up new information and converges on the ideal parameters.

2. Number of Dense and Convolutional Layers: The model has four Dense layers and three Conv2D layers. The kernel sizes for the convolutional layers are (3, 3) and the filter sizes are 128, 64, and 32, respectively. There are 512, 256, 128, and 64 neurons in the dense layers, correspondingly.

3. Dropout Rates (Dropout): Dropout layers with dropout rates of 0.4, 0.3, 0.2, and 0.2 are utilized after each dense layer. These dropout rates represent the percentage of neurons that were randomly turned off during training to avoid overfitting.

4. LeakyReLU and tanh are the two activation functions that were employed. Both have an alpha value of 0.1. LeakyReLU introduces non-linearity and permits modest negative values to avoid training with dead neurons. Tanh is used in some deep layers as an activation function to add non-linearity to the learning process of the model.

Aspect	**Value**
Learning Rate	0.001
Number of Convolutional	3
Number of Dense Layers	4
Dropout Rates	0.4, 0.3, 0.2, 0.2
Neurons in Dense layers	512, 256,128,64
Activation Functions	LeakyReLU (alpha=0.1), tanh

B. Testing with anonymous data:

The model performs a sequence of convolutional, activation, pooling, and dense operations to extract and analyze pertinent features from an input image with the dimensions (256, 256, 3). The spatial dimensions of the feature maps are further condensed by the GlobalAveragePooling2D layer.

The softmax activation function is used at the 11 neurons that make up the last Dense layer. The softmax function creates a probability distribution over the 11 classes by converting the raw output values into probabilities. The model's confidence score (probability) for each value in the output corresponds to the relevant class.

For instance, when a given input image, such as a rain image, results in an output probability distribution of [0.02, 0.15, 0.10, 0.01, 0.05, 0.50, 0.03, 0.08, 0.04, 0.01, 0.01] for the given CNN architecture, the model is most confident that the image belongs to class 6, which has the highest probability (0.50). As a result, the model labels the input image as "Class 6," which probably refers to a category of wet weather.

In order to classify rainy photos, Initial Conv2D layers are used to identify rain-related edges and patterns, such as raindrops and water streaks. This technique for testing may be quickly explained. Intricate traits that mix regional structures, like water reflections, are isolated in deeper layers. By identifying global patterns, GlobalAveragePooling2D lowers sensitivity to particular regions. In order for the model to learn class-specific discriminative representations, dense layers sharpen features. Softmax activation generates a probability distribution that yields confidence scores for each class. The system chooses the most likely class label, such as "Class 6" suggesting wet weather, based on the highest confidence score. The Class 6 classification was given a high likelihood, which means the program found several elements in the rain image that are typical of rainy weather. Because of this multistep process, CNN can categorize wet images with precision and sturdiness.

C. Data Augmentation:

In this weather classification experiment, data augmentation and preprocessing were crucial in increasing the diversity of the training dataset and the generalizability of the model. The main goal was to build a strong model that could properly categorize unseen weather photographs from the dataset, which included images reflecting different weather conditions. A number of data augmentation approaches were used to accomplish this. To provide uniform scale across all photos for improved convergence during training, the pixel values of the images were first rescaled to a normalized range between 0 and 1. The next step involved horizontal flipping, which introduced different orientations for weather patterns and enabled the model to identify weather conditions regardless of their left-to-right orientation.

Additionally, a random rotation with a maximum angle of 20 degrees was used, providing a variety of perspectives on weather conditions and preparing the model to handle various orientations and angles frequently seen in real-world weather photographs. By randomly zooming in or out on the photographs with a 15% zoom range, more variation was provided, allowing the model to capture varied degrees of information and improving its adaptability to different image scales and distances. Additionally, up to 50% of the image's width and height were randomly shifted, imitating spatial translations and the variability in weather pattern placement seen in real-world images.

Additionally, the brightness of the photos was randomized within a range of [0.25, 1.5] to allow for various lighting circumstances. This improvement made the model more flexible to real-world circumstances by ensuring that it could recognize weather elements in various illumination conditions. The dataset was then divided into a 20% validation set and an 80% training set with the introduction of a validation split. This separation made it possible to assess the model's performance on hypothetical data, hence reducing overfitting problems during training.

The weather classification model was exposed to a dataset of 5,493 photos over 11 weather classes, with an additional 1,369 images in the validation set, through the use of data augmentation and preprocessing approaches. The model's capacity to generalize to novel and uncharted weather patterns was greatly improved by learning from a wide variety of weather scenarios, orientations, scales, and lighting conditions. As a result, the model showed enhanced precision and durability, making it a highly efficient and dependable tool for correctly categorizing meteorological conditions in real-world scenarios.

V. RESULTS

A. Training Results:

According to the evaluation accuracy results, the model's accuracy on the training data (Epoch 11) and the separate validation dataset was respectively 65.88% and 72.24%. These findings imply that the model is developing during training and generalizing to new data fairly well and the best performance observed in the entire training process was about 72% accuracy on validation data.

```
Epoch 11/12
172/172 [==============================] - ETA: 0s - loss: 1.0250 - accuracy: 0.6588
Epoch 11: val_accuracy improved from 0.66326 to 0.72243, saving model to weather_classifier.h5
172/172 [==============================] - 130s 756ms/step - loss: 1.0250 - accuracy: 0.6588 - val_loss: 0.8794 - val_accuracy: 0.7224
```

B. Testing Results:

The pictures shown below were taken from another dataset. Five different weather classes are represented in the Multiclass Weather Dataset, including cloudy, foggy, rainy, shine, and sunrise. These pictures were gathered from kaggle. On a scale of 0 to 4, where 0 denotes cloudy conditions, 1 denotes foggy conditions, 2 denotes rainy conditions, 3 denotes shine conditions, and 4 denotes sunrise conditions, the photographs are labeled with their relevant meteorological conditions. Each photograph has been categorized into one of the five weather categories according to this labeling. Only the rainy and foggy photos from the aforementioned dataset were utilized to test how well the model performed.

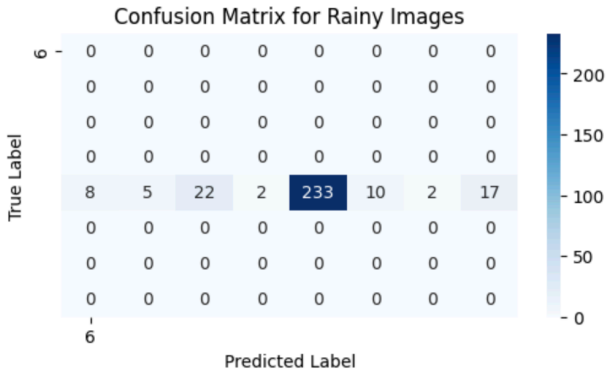
```
1/1 [==============================] - 0s 34ms/step
'rain'
```

```
1/1 [==============================] - 0s 34ms/step
'rain'
```

The outcome demonstrates that 233 of the 300 photos in the rainy class were accurately identified as such by the model. This indicates that the model properly classified photos of precipitation with an accuracy of about 80%. The accuracy is determined by multiplying the number of photographs successfully classified (233) by the total number of images in the rainy class (300), and then converting that result to a percentage by 100. The algorithm performs better at classifying photographs of rainy weather from the input dataset as accuracy increases.

```
10/10 [==============================] - 1s 90ms/step
Predicted class labels for rainy images:
[ 6  6  6  6  6  6  6  6  6  8  6  6  6 10  4  4  6  6  6  6  8  6  6
  6  6  6  6  6  4 10  6  6  6  4  6  8  1  6  6  6  6  6  6  1  6  6  6
  6  6  6  6  4  6  6  6  6  6  6  6  6  6  6  6  6  9  3  6  6  8  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6 10
  6  6  6  6  4  6  6  4  6  6  4  6  6  6  4  6  6  1  6  6  6  6  6  6
  6  6 10  6  6  6  4  6  5  6  6  6  6  6  6 10  6  4  6  6  6  6  4  6
  6  6  6  6  4  6  6  6  6  6  4  6  6  6  6  6  6  6  6  6  6 10  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6 10  6  6  6  6  6  6  6  6  9 10  6  6  6  6  6  4  6  6  6  8  3  6
  6  6  8  6  6  6  6  6 10  6  1  6 10  5  6  1  6  6  6 10  6 10  6  6
  6  4  6  6  6  3  6  8  4  4  6  6  6  6 10  4  6 10  8  6  6  6 10  3
  6  4  6  6  6  8  1  4  6  6  6  6  6  8  1 10  6  6  3 10  1  6  6  6
  6  6  6  6  4  6  6  6  6  6  6]
```

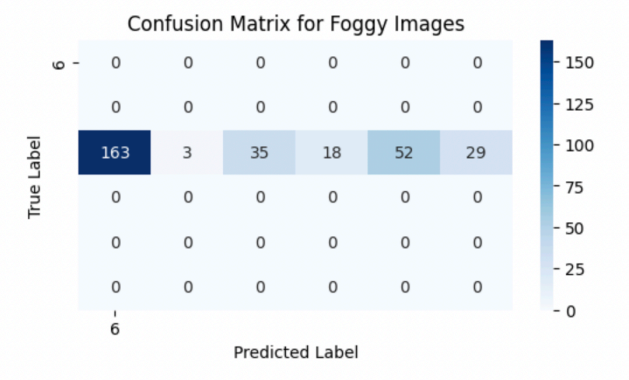
Confusion matrix for rainy images:



Foggy data:

The results show that the model correctly classified 163 out of the 300 images in the foggy class. This shows that the algorithm correctly identified photographs taken in foggy conditions with an accuracy of about 54.33% (about 60%).

```
10/10 [==============================] - 1s 91ms/step
Predicted class labels for foggy images:
[ 8  9  9  1  8 10  9  1  1 10  1  1  1  1  1 10  9  1  6  1  8  1 10  1
  6  1  6  1  1  1  1  6  9  6  1  1  9 10  6  1  1  6  9  6  1  1  6  8
  8 10  1  1  9  1  1  9  1  1  6  9  1  9  1  1  1  5  1  6  1  9  8  8
  9  5  1  6  1  1  1  1  1  1  6  6  1  9  1  6  1  1  9  9  1  1  1 10
  9  1  6  9  1  1  1 10  1  1 10  9  8 10  1 10  9  6  1  1  1  1 10  9
  1  1  9  9  1  1  1  1  9 10  1  1  1  8  8  8  1  9  9 10  1  1  1  1
 10  1  6  1 10  1  9  9 10  9  1  1  6 10  9  9  1  1  1  9  1  1  1  1
  1 10  1 10  6  1  1  1  9  1  1  1  1  1  9  1  1  9  1  1  6  9 10  1
 10  1  1  1  8  1 10  1  1  6  8  1  1  5  1  1 10  1  1  9  8 10  1  9
  1  1  6  6  9  1  1  1  1  1  1  1  1  6  1  8  1  6  1  1  9  1  6  1
  1  6 10  6  9  1 10  1  1  9  6  1  8 10  1  1  6  1  9  1  8  1  1  1
  9  9  1  1  1  6  1 10  1  8  1  9  6  1  1  1  1  1  9  1  6  1  1  9
  1  1  9  1  6  1  1  9  1  9  1  9]
```



VI. CONCLUSION

Convolutional Neural Network (CNN) technology was used to create this model, which would classify weather photos into 11 categories. A relatively small dataset of roughly 6000 weather photos from various sources served as its training data. The model performs well despite the sparse data, with a training accuracy of roughly 66% and a validation accuracy of about 73%. The ability of this model to attain an astonishing 80% accuracy on real-time test data, which significantly differs from the training data, is what truly sets it apart from other models. This implies that the model generalizes well to hypothetical weather photos, demonstrating its sturdiness and flexibility.

In the future, we aim for substantially expanding the training dataset to solve any potential restrictions brought on by its short size. The model attempts to increase its comprehension of complex patterns and fluctuations in weather conditions, resulting to higher performance on both the training and test data, by adding a wider and more varied set of weather photos. This improvement will make the model much more efficient and dependable for classifying weather images, opening the door for its use in a wider variety of real-world applications.

REFERENCES

[1] Zhao, B., Li, X., Lu, X., & Wang, Z. (2018). A CNN-RNN architecture for multi-label weather recognition. Neurocomputing, 322, 47-57.

[2] Xie, K., Huang, L, Zhang, W., Gin, Q., & Lyu, L. (2022). A CNN-based multi-task framework for weather recognition with multi-scale weather cues. Expert Systems with Applications, 198, 116689.

[3] X. Zhao and C. Wu, "Weather Classification Based On Convolutional Neural Networks," 2021 International Conference on Wireless Communications and Smart Grid (ICWCSG), Hangzhou, China, 2021, pp. 293-296, doi: 10.1109/ICWCSG53609.2021.00064.

[4] M. Elhoseiny, S. Huang, and A. Elgammal, "Weather classification with deep convolutional neural networks," International Conference on Image Processing (ICIP), IEEE, Canada, 2015.x

[5] Zhang, Z., Ma, H., Fu, H., & Zhang, C. (2016). Scene-free multi-class weather classification on single images. Neurocomputing, 207, 365-373.

[6] Zhao, B., Hua, L., Li, X., Lu, X., & Wang, Z. (2019). Weather recognition via classification labels and weather-cue maps. Pattern Recognition, 95, 272-284.

[7] Guerra, J. C. V., Khanam, Z., Ehsan, S., Stolkin, R., & McDonald-Maier, K. (2018, August). Weather Classification: A new multi-class dataset, data augmentation approach and comprehensive evaluations of Convolutional Neural Networks. In 2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS) (pp. 305-310). IEEE.

[8] Chen, L. C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587.