Weather Recognition Using Self-supervised Deep Learning

Diego Acuña-Escobar[1], Monserrate Intriago-Pazmiño[1](✉), and Julio Ibarra-Fiallo[2]

[1] Departamento de Informática y Ciencias de la Computación, Escuela Politécnica Nacional, Quito, Ecuador
{diego.acuna,monserrate.intriago}@epn.edu.ec

[2] Colegio de Ciencias e Ingenierías, Universidad San Francisco de Quito, Cumbayá, Ecuador
jibarra@usfq.edu.ec

Abstract. The automatic recognition of weather in images has many important applications in different fields, such as: land and air traffic control, autonomous vehicles, road safety warnings, crop control, improvement of images taken in outdoor areas, among others. Despite the great applicability, this field of study has not yet been explored in detail, primarily due to the great challenge and difficulty involved in extracting deterministic features for each type of weather. Several works have focused their efforts on designing binary classifiers that allow determining just two classes. A difficulty lies especially in the fact that the target classes are not completely exclusive in an image. Different classes can share the same features. Another difficulty that previous work has faced is the need for a large number of labeled images to model the various weather states. In this work, we propose an approach called self-supervised deep learning applied to weather recognition in order to reduce the requirement of the huge amount of labeled images. Our architecture, a ResNet-50 implementation, is responsible for obtaining the representations of each unlabeled image with a self-supervised approach for both pre-training and fine-tuning steps. It has been used transfer learning for sharing the architecture between these steps. Our results reached an average accuracy of 0.8833. Based on this result, it can be concluded that self-supervised learning is a convenient solution to obtain high performance in the weather recognition task from digital images.

Keywords: Weather recognition · Self-supervised deep learning · Residual learning · Transfer learning · Fine tuning

1 Introduction

The weather's state is one of the most important variables to be considered when deciding on doing one activity or another. It is so important that it can even influence our mood and the consequences that derive from it. Knowing the different weather states in relation to the time has allowed to determine the biodiversity of species, ecosystems, and natural places generated from them. Places can be habitable or completely uninhabitable due to the weather [1]. Technological advances have initially allowed the design of analytical systems that seek to predict the state of the weather based on historical information on the behavior of the weather over time using variables such as temperature, atmospheric pressure, winds, humidity, and precipitation.

Currently, with the advancement of machine learning for automatic image processing, it has been possible to model the state of the weather based on graphical data extracted from training images containing the different weather states: cloudy, foggy, rainy, shine and sunrise [18]. These models have allowed solving problems ranging from approaches as simple as walking or riding a bicycle in a city to more complex solutions such as autonomous driving assistants [8].

According to the authors in [19], rainstorms, blizzards, and fog are three kinds of the most studied extreme weather. Figure 1 shows four types of extreme weather conditions from the Multi-class Weather Dataset(MWD) [18], which will lead to reduced visibility and friction coefficient of road, resulting in tremendous potential dangers. For that reason, automatically recognizing weather is essential for many applications, such as highway traffic condition warnings, automobile auxiliary driving, climate analysis, and so on.

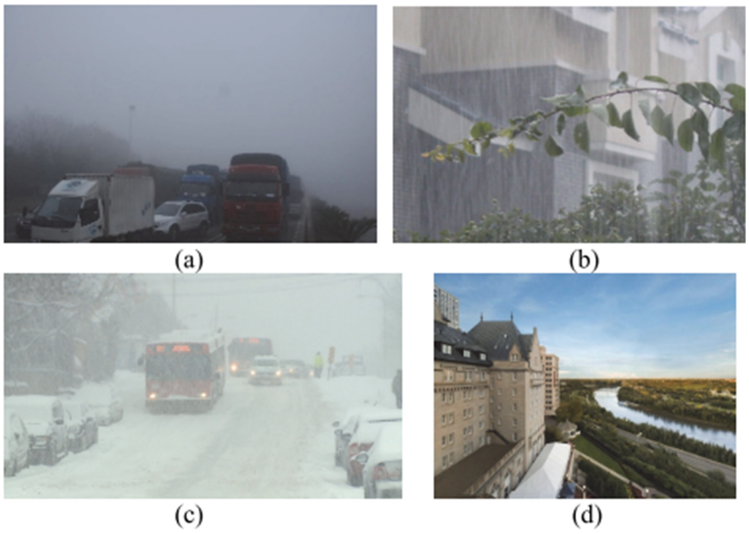


Fig. 1. Extreme weather conditions from the MWI dataset [18]

Many approaches and methodologies have been proposed in the field of weather recognition, like multitask learning [9], dictionary and multiple kernel learning [18], convolutional neural networks, and others. All these approaches have a problem balancing the efficiency of the solution and the number of images required to train the algorithm in a supervised model.

Training a model with a fully deep learning supervised approach requires a synergistic effort to obtain adequate and especially correctly labeled training images that allow the algorithm to learn. The training images require the maximum amount of, making it very difficult to apply deep learning when labeled data is scarce, as in the case of weather recognition. It varies by case, but in most cases, training a deep learning model requires thousands, hundreds of thousands, millions, even billions of training images to learn accurate representations of the images [7].

Once the training images difficulty is solved, another common problem in Deep Learning approaches is the well known *vanishing/exploding gradients*, which is a cause of increasing the depth of a deep learning model. It is essential to understand that the network depth is of crucial importance. The leading results in the ImageNet challenge [12] implement "*very deep*" models with a depth of sixteen to thirty stacked layers [16]. The degradation problem is attributed to an increased depth while the accuracy gets saturated. This is not caused by overfitting, and while adding more layers the training and test errors get higher (See Fig. 2).

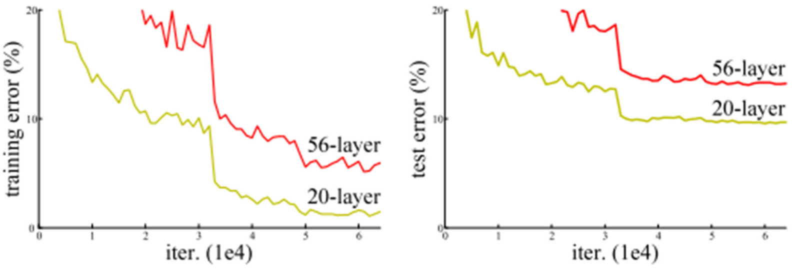


Fig. 2. Training and test errors in plain deep networks [6]

In this work, we will deal with the weather recognition problem focused on five weather types: cloudy, foggy, rainy, shine and sunrise. To extract relevant information from the training images, we follow a "very deep" neural network architecture following a ResNet-50 implementation for a self-supervised pre-training and a supervised fine tuning.

The rest of this paper is organized as follows. Section 2 provides fundamental details of related works. In Sect. 3, the datasets and the method are described. Next, results, a comparison with other works, and a discussion are stated in Sect. 4. Finally, some conclusions are presented in Sect. 5.

2 Related Works

In [18], the authors propose a method to classify the weather among sunny, rainy, snowy, and haze images. The method is based on multiple weather features, learning dictionaries, and kernel learning algorithm. Sky, shadow, rain streak, snowflake, and dark channel are extracted as local characteristics. These features are processed using several algorithms, for example, shadow and rain are represented using Histogram of Gradient (HOG), snowflake is described as a kind of noise. Then, multi-feature and class specific dictionaries are created. However, the dictionaries are shared for all weather classes. Finally, the decision is performed by feature fusion. The multiple kernel learning approach is used to obtain the best weights for all features. The proposal was evaluated using their own public dataset, Multi-class Weather Image (MWI), which is composed of 20K images. The method performance was 0.7139 on the accuracy average.

Images can also be associated with other rich image-weather association data, like temperature and humidity. In [3], the authors associate visual data with heterogeneous metadata to build a more robust weather classifier to estimate weather properties from single images. The authors target the properties: weather types (sunny, cloudy,snowy, rainy, and foggy), temperature (between $-25\,^{\circ}$C and $45\,^{\circ}$C), and humidity (between 0% and 100%). Regarding weather types, the proposal computes several features and creates a random forest classifier. This proposal was trained and tested using Image2Weather dataset. It is a public dataset, and the whole targets obtained in this work are included on its website. The results report 0.766 on average accuracy classifying weather types. Other interesting work presented in [8], achieved an accuracy of almost 90%, training a CNN model with the Image2Weather dataset which consists of more than 180000 images of global landmarks of four weather categories, such as sunny, cloudy, rainy, snowy, and foggy. They introduce a framework of parallel deep CNN models to recognize weather and visual conditions from street-level images of urban scenes using four deep CNN models to detect dawn/dusk, day, night-time, glare, rain, snow, and fog.

The implemented models refer to: 1) NightNet detects the differences between dawn/dusk, day and night-time. It aims to understand the subtleties of street-level images despite the dynamics of weather conditions and urban structure, 2) GlareNet detects images with glare regardless of its source (sun or artificial light) for both dawn/dusk, day and night-time of various weather conditions.

Different network architectures have been proposed to face image recognition in general tasks. They have been divided into two general groups: plain and residual architectures. As plain network implementation, we can mention the VGG nets [15] with convolutional layers mostly of 3×3 filters. These implementations follow two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled to preserve the time complexity per layer. Figure 3 illustrates the general groups for different network architectures.

Residual Learning methods have achieved the highest results in the Imagenet Dataset challenge [12]. Residual Learning asymptotically approximates residual functions in the form, $H(x) - x$, where x denotes the inputs in the first layer and $H(x)$ represents the underlying mapping function for some stacked layers. So rather than approximate $H(x)$, this layers will approximate $F(x) := H(x) - x$. The original function thus becomes $F(x) + x(1)$. This way, if adding more layers as identity mappings, a deeper model should have a training error no greater than its shallower counterpart.

In experiments, Fig. 4, it has been shown that the learned residual functions in general have small responses. It suggests that identity mappings provide reasonable preconditioning.

$$y = F(x, Wi) + x \tag{1}$$

To reduce the amount of labeled samples, a method called self-supervision has been proposed, which is one of the most future promising frameworks that improve the accuracy of the prediction models, not only for image recognition tasks, also for time-series signals recognition [13]. For image processing, in [4], it was achieved 85% top-1 accuracy by using only the 10% of the Imagenet data. Common image transformations or corruptions (see Fig. 5) have been applied to generate the auto labeled data [11], like: Gaussian Noise, Shot Noise, Impulse Noise, Defocus Blur, Frosted Glass Blur, Motion Blur, Zoom Blur, Snow, Frost, Fog, Brightness, Contrast, Elastic, and Pixelate.

These transformations have been tested for image classifier robustness. It standardizes and expands the corruption robustness topic while showing which classifiers are preferable in safety-critical applications. It also evaluates performance on common corruptions and perturbations, not worst-case adversarial perturbations.

Self-supervised learning has also been proved to be successful in other critical tasks like medical image classifications. The authors in [2], introduce a novel Multi-Instance Contrastive Learning framework that uses multiple images of the underlying pathology per patient case for medical image classification. In this work, three steps are proposed: (1) supervised pretraining on a large labeled dataset such as ImageNet. (2) self-supervised pretraining using contrastive learning on unlabeled data. (3) Supervised fine-tuning on labeled medical images.

Finally, the study of transfer learning (see Fig. 6) also assumes significance for this study as it is motivated by the fact that people can intelligently apply knowledge learned previously to solve new problems faster or with better solutions [10].

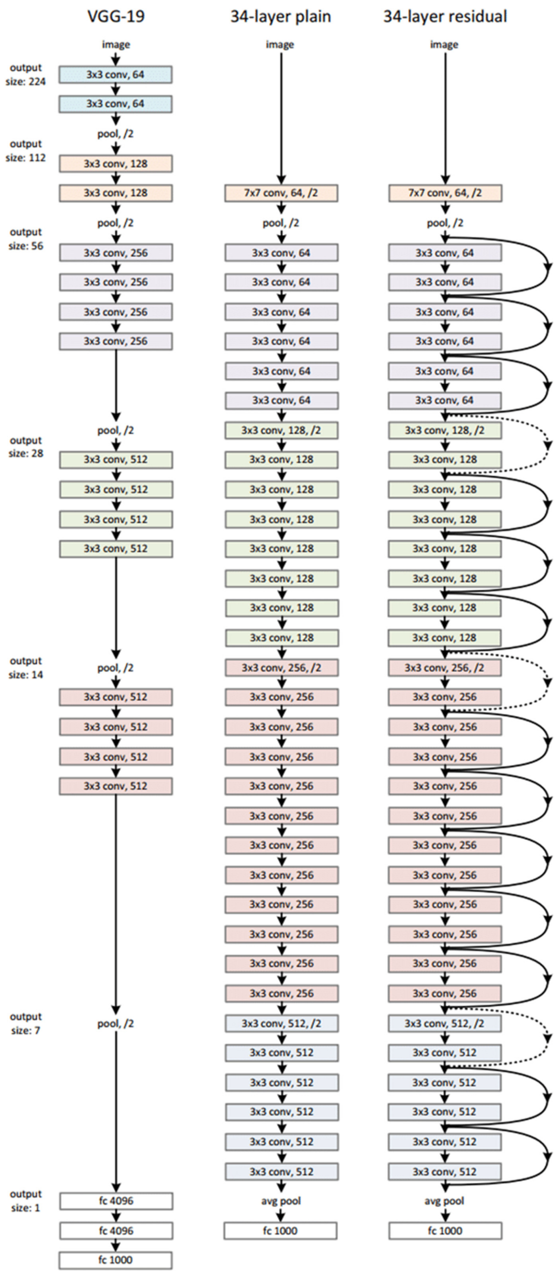


Fig. 3. Example network implementations for image recognition [6]

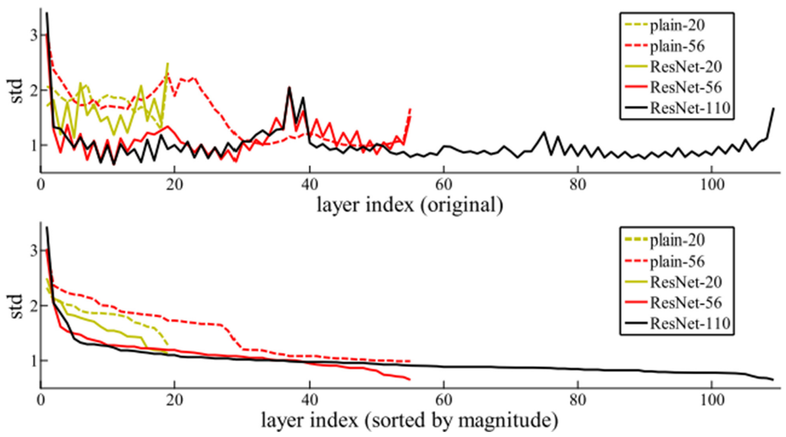


Fig. 4. Standard deviations(std) of layer responses on CIFAR-10 [6].

3 Materials and Method

3.1 Dataset

The data was extracted from the public dataset called weather dataset [5], which is available and was used for other weather recognition studies. This data was analyzed, pre-processed, and then fed to the artificial neural network for the self-supervised pre-training task. This dataset contains 300 training images for each class: cloudy, foggy, rainy, shine, and sunrise, along with other 30 images for testing and validation. These images are in different sizes and utilize the RGB model for the color description. The images need to be resized to an input size of $224 \times 224 \times 3$ to have the required size for the ResNet architecture. These images will be applied to common transformations as shown in Fig. 5 in order to build a binary classifier model. In the first step, the model must learn to distinguish if an image was previous transformed (class 1) or not transformed (class 0). To discriminate between these two classes, the model must extract some general features of each image. In this step, it does not matter which real class the image belongs to. It is just a pre training step.

Once the self supervised model is pre-trained, it needs a fine tuning process for which we used other random images from a different weather public dataset called Multi-Class Images for Weather Classification [14]. From this dataset, we extracted 200 random images for each class for the fine tuning and 100 other random were used for validation.

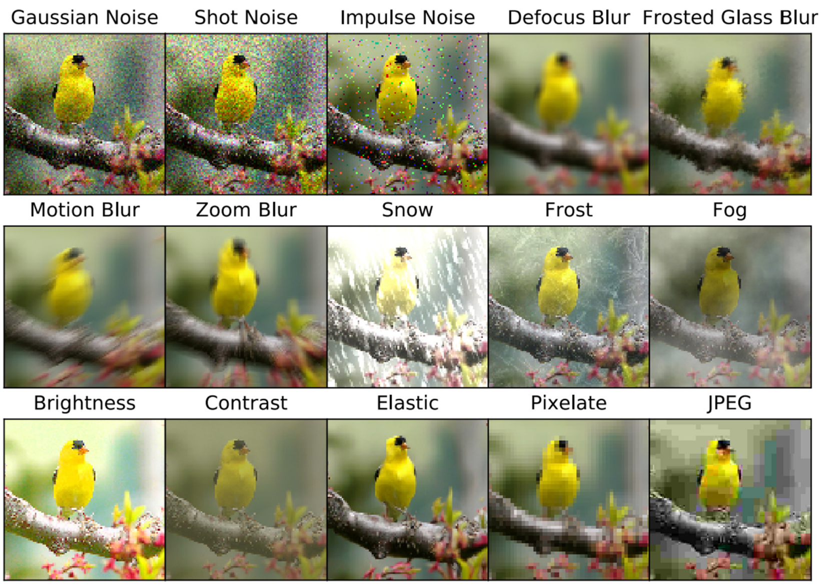


Fig. 5. Common image transformations [11]

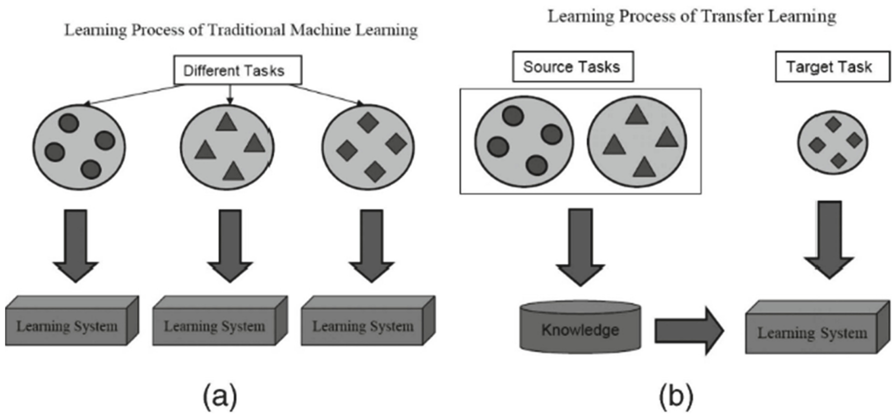


Fig. 6. Traditional machine vs transfer learning

3.2 Method

Figure 7, shows a general overview of our proposed method, which is composed by three main features: pre-training, transfer learning and fine tuning.

We implemented a residual learning architecture with 50 layers depth as suggested in [6]. Residual learning minimizes the gradient vanishing problem which is common while training. Deep neural networks uses shortcuts connections or identity mappings to connect the features in the building blocks (see Fig. 8a).

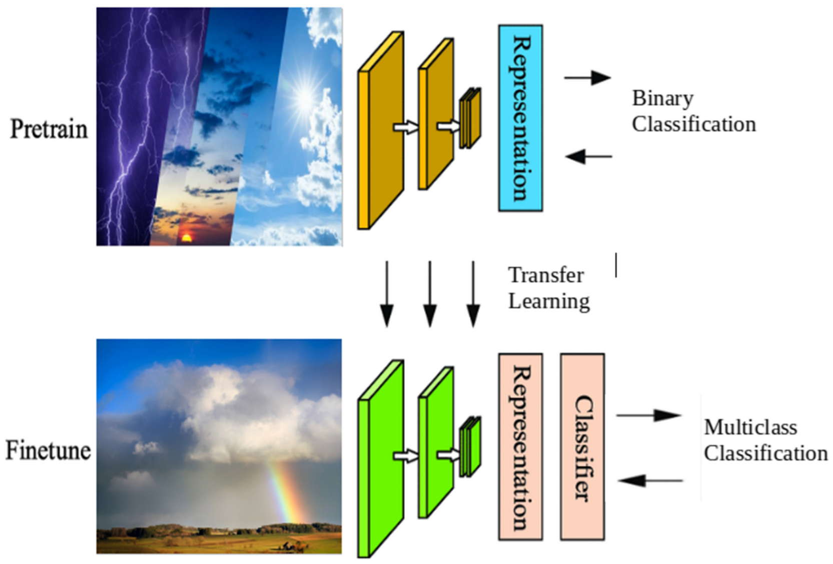


Fig. 7. General overview of the proposed method

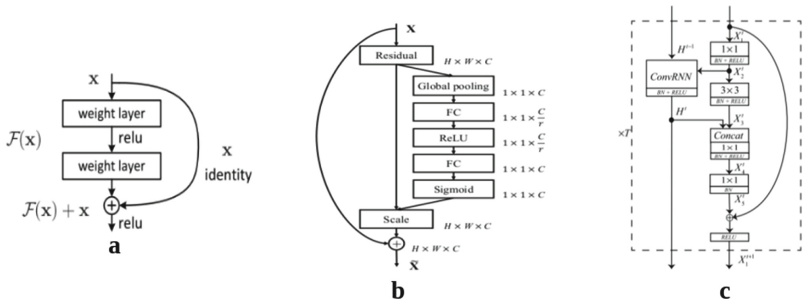
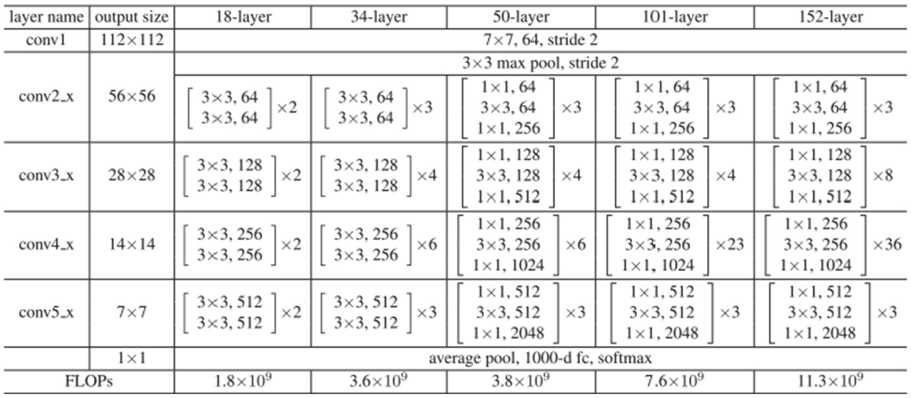


Fig. 8. Residual learning: a) building block [6] b) SE-ResNet block [7] c) RegNet block [17]

One variation made to the default ResNet architecture is adding Squeeze and excitation blocks (SE) [7], which aim to improve performance and increase model complexity. SE blocks adaptively recalibrate channel-wise feature responses by explicitly modeling interdependencies between channels (see Fig. 8 b). Finally, we added a regularization module [17], to capture the spatio-temporal dependency between building blocks while constraining the speed of parameter increasing (see Fig. 8).

For each residual function, in the ResNet 50 implementation (see Fig. 9), it is used a stack of 3 layers. The three layers are 1×1, 3×3, and 1×1 convolutions, where the 1×1 layers are responsible for reducing and then increasing dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 9. ResNet architectures [6]

Our proposed method includes three stages: preprocessing, pre-training, and fine tuning.

Preprocessing. All the images in the dataset have different sizes, and each of them was cropped randomly to a size of $224 \times 224 \times 3$ to fit the input layer size. At the same time, the images were transformed and tagged accordingly. Labels transformed and not transformed were used in this step. Finally, all the images were saved in a single file with extension $.h5$. Figure 10 shows some transformations applied to all the images for the self-supervision pre-training.

Pre-training. Pre training a self-supervised model includes using the image common transformations [11]. It allows the model to learn and predict if an image is transformed or if it is not. This step is implemented as a binary classification. Image is transformed class 1 and image is not transformed class 0. Here we use the ResNet-50 architecture implemented. Figure 11 shows the architecture and the results for the binary classification pre-training.

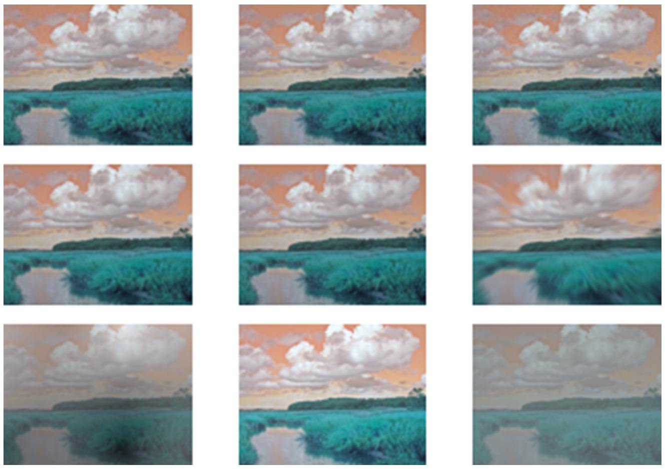


Fig. 10. Transformations applied

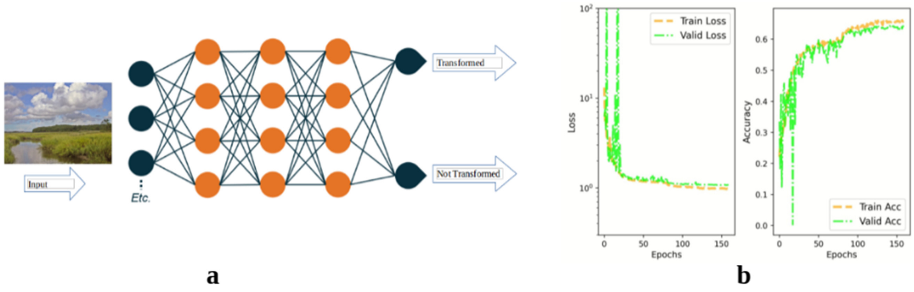


Fig. 11. a) Pre training network. b) Pre training process

Fine Tuning. We fine tuned the model using 200 random labeled images for each weather class. Transfer learning is used at this stage, so we can reuse part of the pre-trained model and just change the dense layer for using the real weather classes. Figure 12 shows the process for the fine tuning.

4 Results and Discussion

The model was implemented in python, and using the TensorFlow machine learning framework from Google. It was tested in three different hardware for getting the best hyperparameters such as: epochs, batch size, learning rate, performance metrics, and execution time. In most cases, it took three days for pre-training the model and six hours for the fine tuning.

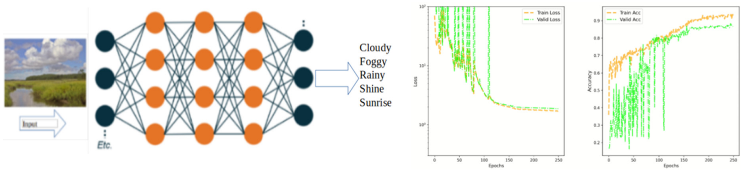


Fig. 12. Fine tuning process

- 64 GB ram, i7 cpu, Ubuntu.
- 32 GB ram, 8 GB GPU, i7 cpu 4 vCPUs, Windows 10.
- azure virtual machine: 32 GB ram, 4 vcpus, Ubuntu.

We used the confusion matrix (see Fig. 13), to demonstrate the results obtained by our proposed method in the validation predictions. Then, we calculate precision, recall and F1 to measure the performance of our algorithm, see Table 1.

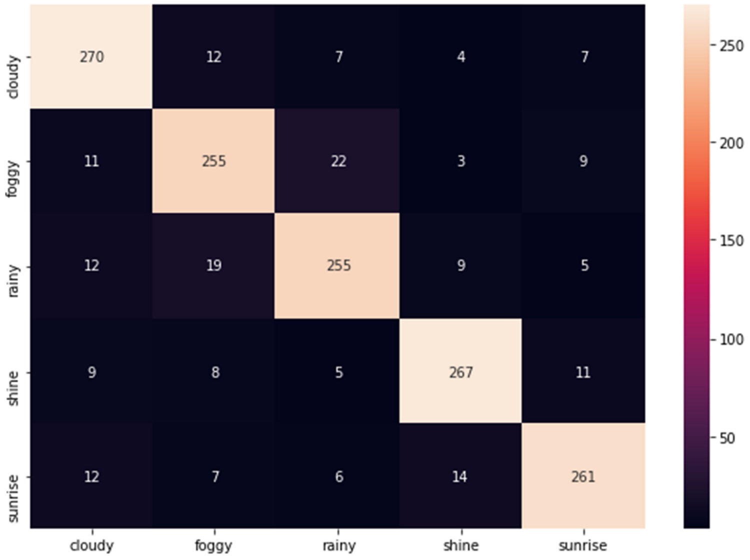


Fig. 13. Validation confusion matrix

The experiments achieved an average accuracy of 88%. It is graphically reported in Fig. 12. Table 2 shows a comparison with other related works that have been detailed in a previous section.

As shown below, the results obtained by our proposed method improve the results obtained by other related works that focused on fully supervised methods, considering that we drastically reduced the number of labeled images used in the

Table 1. Metrics obtained by our experiments

	Cloudy	Foggy	Rainy	Shine	Sunrise
Precision	09	0.85	0.85	0.89	0.87
Recall	0.86	0.85	0.86	0.9	0.89
F1	0.88	0.85	0.86	0.86	0.88

Table 2. Performance of the proposed method and other works

Method	Year	Classes	Accuracy
[18]	2016	Sunny, rainy, snowy, haze	0.7139
[3]	2017	Sunny, cloudy, snowy, rainy, foggy	0.7660
Proposed self-supervised	2021	Cloudy, foggy, rainy, shine, sunrise	**0.8833**

training stage and were faced with a great limitation in computational resources. Our results can be further improved by solving the hardware issues to be able to do a pre-training with a multi-class approach instead of adopting a binary classification.

5 Conclusions

In this research work, a convolutional neural network has been implemented with the addition that it is not fully supervised and it introduces self-supervised learning.

Experiments showed that pre-training a model with self-supervised learning can help achieve better results compared with some related works about weather recognition with fully supervised training.

The results achieved are promising. However, it is necessary to recognize that more computational resources are fundamental in order to derive a convenient model. It is really important to plan the resource requirements when training with big images since it might require high physical memory availability.

The network architecture should consider the size of the images and the computational resources available to get good training and validation results.

In view of the above, there are several suggestions for future works. Among them, it is required that the model grows and self-learns using a greater number of rotations. Images can be included from other datasets in the self-supervised learning phase to find a more generalized solution and to evaluate the model with a higher number of images.

References

1. Why is weather important in people's lives?. http://scienceline.ucsb.edu/getkey.php?key=4580

2. Azizi, S., et al.: Big Self-Supervised Models Advance Medical Image Classification, January 2021. https://arxiv.org/abs/2101.05224v2
3. Chu, W.T., Zheng, X.Y., Ding, D.S.: Camera as weather sensor: estimating weather information from single images. J. Vis. Commun. Image Representation **46**, 233–249 (2017). https://doi.org/10.1016/j.jvcir.2017.04.002
4. Goyal, P., et al.: Self-supervised pretraining of visual features in the wild, Technical report (2021)
5. Gupta, V.: Weather dataset—Kaggle. https://www.kaggle.com/vijaygiitk/multiclass-weather-dataset
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, December 2016, pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
7. Hu, J., Shen, L., Albanie, S., Sun, G., Wu, E.: Squeeze-and-excitation networks. IEEE Trans. Pattern Anal. Mach. Intell. **42**(8), 2011–2023 (2020). https://doi.org/10.1109/TPAMI.2019.2913372
8. Ibrahim, M.R., Haworth, J., Cheng, T.: WeatherNet: recognising weather and visual conditions from street-level images using deep residual learning. ISPRS Int. J. Geo-Inf. **8**(12), 549 (2019). https://doi.org/10.3390/IJGI8120549. https://www.mdpi.com/2220-9964/8/12/549/htmwww.mdpi.com/2220-9964/8/12/549
9. Li, X., Wang, Z., Lu, X.: A multi-task framework for weather recognition. In: Proceedings of the 2017 ACM Multimedia Conference, MM 2017, pp. 1318–1326 (2017). https://doi.org/10.1145/3123266.3123382
10. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010). https://doi.org/10.1109/TKDE.2009.191
11. Rudy, J., Ding, W.G., Im, D.J., Taylor, G.W.: Benchmarking neural network robustness to common corruptions and perturbations, pp. 1–9 (2015)
12. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015). https://doi.org/10.1007/S11263-015-0816-Y
13. Saeed, A., Ozcelebi, T., Lukkien, J.: Multi-task self-supervised learning for human activity detection. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. **3**(2), 1–30 (2019). https://doi.org/10.1145/3328932
14. Sharma, S.: Multi-Class Images for Weather Classification—Kaggle. https://www.kaggle.com/somesh24/multiclass-images-for-weather-classification
15. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, September 2014. https://arxiv.org/abs/1409.1556v6
16. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway Networks, May 2015. https://arxiv.org/abs/1505.00387v2
17. Xu, J., Pan, Y., Pan, X., Hoi, S., Yi, Z., Xu, Z.: RegNet: self-regulated network for image classification, pp. 1–6 (2021). http://arxiv.org/abs/2101.00590
18. Zhang, Z., Ma, H., Fu, H., Zhang, C.: Scene-free multi-class weather classification on single images (2016). https://doi.org/10.1016/j.neucom.2016.05.015
19. Zhu, Z., Zhuo, L., Qu, P., Zhou, K., Zhang, J.: Extreme weather recognition using convolutional neural networks. In: Proceedings - 2016 IEEE International Symposium on Multimedia, ISM 2016, pp. 621–625 (2017). https://doi.org/10.1109/ISM.2016.81