**AUTOMATED WEATHER CLASSIFICATION USING TRANSFER LEARNING - InceptionV3, MobileNetV2 and ResNet50**

**A MINI PROJECT REPORT**

Submitted by

**JANANI T 61072012110**

**JUHIKAUSAR S 61072012112**

**REKHA S 61072012132**

**SWATHI A 61072012141**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**GOVERNMENT COLLEGE OF ENGINEERING-BARGUR (AUTONOMOUS)**

**KRISHNAGIRI-635 104,TAMILNADU.**

(An Autonomous Institution and Affiliated to Anna University)

**ABSTRACT**

Weather recognition is a common problem for many branches of industry. For example self-driving cars need to precisely evaluate weather in order to adjust their driving style. Modern agriculture is also based on the analysis of current meteorological conditions. One of the solutions may be a system detecting weather from image. Because any special sensors are needed, the system should be really cheap. Thanks to transfer learning it is possible to create image classification solutions using a small dataset. In this paper three weather recognition models are proposed. These models are based on InceptionV3, MobileNetV2 and ResNet50 architectures. Their efficiency is compared and described.

**Keywords:** Machine learning · Deep learning · Transfer learning · Image classification · Convolutional Neural Networks (CNN) · Neural network architecture · Weather classification

**INTRODUCTION**

# Weather Classification

The most popular methods for the automatic recognition of the current weather is the use of weather stations. They are systems equipped with specialized hardware (e.g. light intensity sensors, rain detectors, temperature sensors, humidity sensors). These systems collect very detailed data that it is reflected by their relatively high cost. A cheaper alternative may be to use the magic of machine learning which allows to build a classifier that will analyze the provided photo and be able to determine the general weather condition. This paper evaluate models built on top of three different neural network architectures.

# Transfer Learning

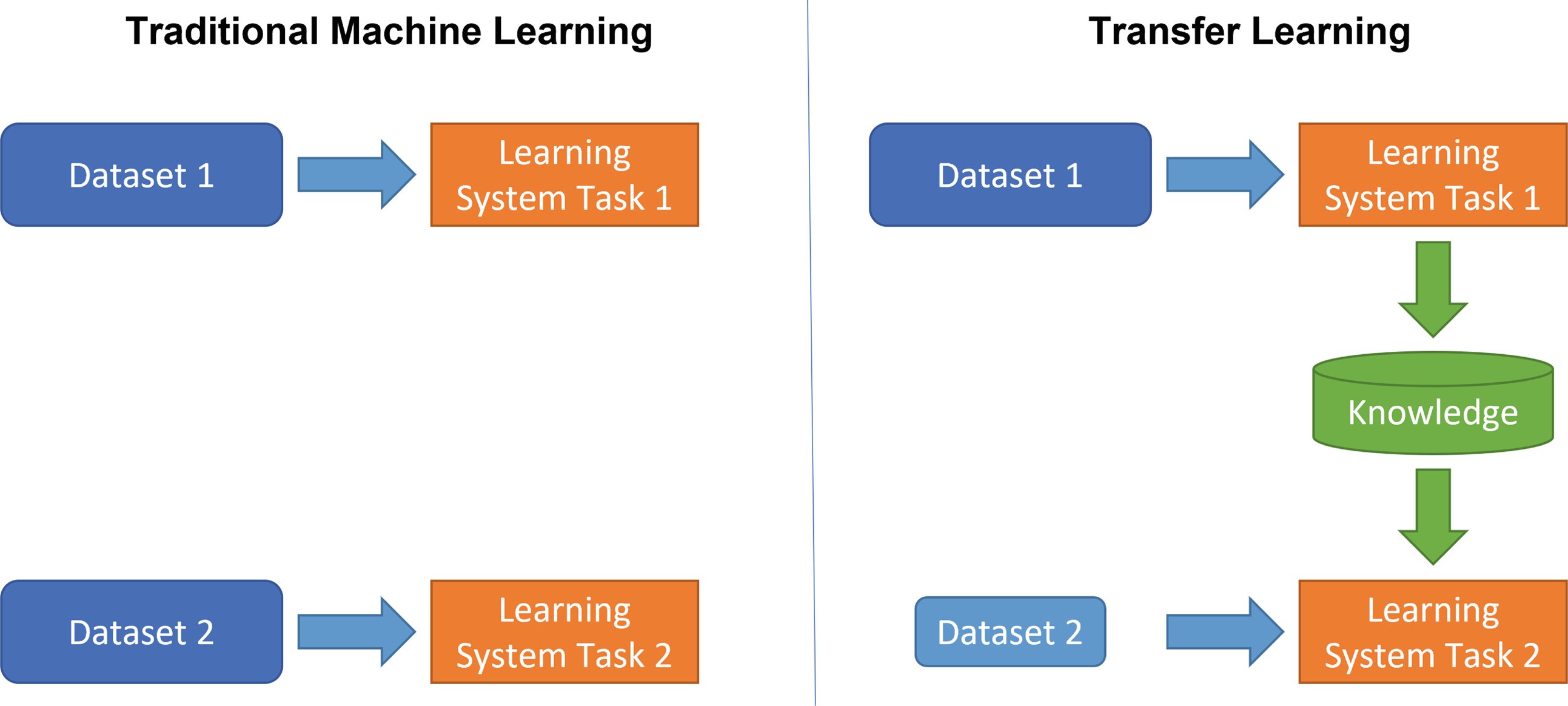
Transfer learning is a machine learning technique of reusing a previously prepared model to train new one for related problem. Instead of starting the learning process from scratch, it starts with the patterns learned in solving the related task. It is widely used in image recognition because neural networks try to detect edges, shapes and then features. With transfer learning it is possible to use edge

c The Author(s) 2022

C. Biele et al. (Eds.): MIDI 2021, LNNS 440, pp. 3–11, 2022.

<https://doi.org/10.1007/978-3-031-11432-8_1>

and shapes detection layers of pre-trained model and then train custom feature layers. It is much faster than training model from scratch, uses less computing power and allows to train model with a relatively small amount of data. Overall concept of transfer learning shows Fig.1.



**Fig.1.** Transfer learning concept

# Architectures

Inception, ResNet, and MobileNet are the convolutional neural networks commonly used for an image classification task. Although they carry out similar problems and are based on different architectures, some differences can be expected in the results of specific tasks such as weather classification.

## Inception

Inception architecture is based on two concepts - 1×1 Convolution and Inception Module. Deep neural networks are expensive in terms of computation. Thanks to 1×1 Convolution it is possible to decrease number of computations by reducing number of input channels. It causes that depth and width of neural network can be increased. Inception Module performs computations of some convolution layers simultaneously and then combines results.

InceptionV3 is a convolutional neural network that is 48 layers deep. The network has an image input size of 299×299.

## MobileNet

MobileNet targets mobile and embedded systems. This architecture is based on an inverted residual structure, which connections are between the bottleneck layers. It uses lightweight depthwise convolutions for features filtering.

This architecture allows to build lightweight models which do not need much computing power.

MobileNetV2 is a convolutional neural network that is 53 layers deep.

The he network has an image input size of 224×224.

## ResNet

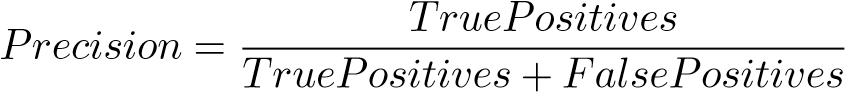
ResNet (Residual Networks) uses concept of identity shortcut connection that allows to jump over some layers. It partially solves vanishing gradients and mitigate accuracy saturation problem. The identity shortcuts simplifies the network and speeds learning process up.

ResNet50 is a convolutional neural network that is 50 layers deep. The network has an image input size of 224×224.

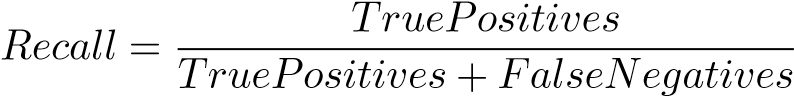
# Metrics

## Precision, Recall, Accuracy, F1

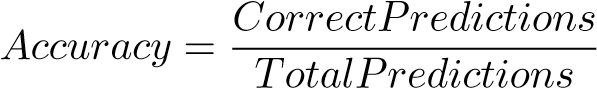
These metrics are widely used in binary classification where only two categories are taken into consideration. In multi classification solutions they might be calculated in multiple ways, but the most popular is to calculate them as the average of every single metric across all classes. Precision represents proportion of predicted positives that are truly positive. Values closer to 1 means high precision and shows that there is a small number of false positives.

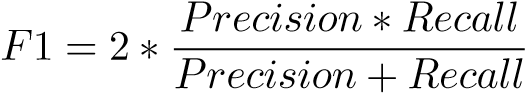


Recall is calculated as a proportion of actual positives that have been classified correctly. Values closer to 1 means high recall and shows that there is a small number of false negatives.



Accuracy measures proportion of number of correct predictions to total number of samples. It helps to detect over-fitting problem (models that overfit have usually an accuracy of 1).



F1 Score combines precision and recall metrics by calculating their harmonic mean. 

## Log-Loss, Log-Loss Reduction

Logarithmic loss quantifies the accuracy of a classifier by penalizing incorrect classifications. This value shows uncertainty of prediction using probability estimates for each class in the dataset. Log-loss increases as the predicted probability diverges from the actual label. Maximizing the accuracy of the classifier causes minimizing this function.

Logarithmic loss reduction (also called reduction in information gain - RIG) gives a measure of how much improves on a model that gives random prediction. Value closer to 1 means a better model.

## Confusion Matrix, Micro-averages, Macro-averages

Confusion matrix contains precision and recall for each class in multi-class classification problem.

A macro-average computes the metric independently for each class and then take the average (treats all classes equally).

A micro-average aggregates the contributions of all classes to compute the average metric.

Micro- and macro-averages may be applied for every metric.

In a multi-class classification problem, micro-average is preferred because there might be class imbalance (significant difference between number of class’ examples).

# Dataset

Models were build on custom six class weather image dataset. Images were scraped from web. Despite the fact that the used trainer does not require data normalization [9], the images were normalized to specified aspect ratio (1:1) and size (512×512 pixels). This image size has been chosen in order to not to favor any architecture (InceptionV3 prefers 299×299, MobileNetV2 and ResNet50 prefer 224×224). Training set details are described below.

Total: 1577 images

Image format: JPEG

Image size: 512×512 pixels

Color space: sRGB Categories:

* Clouds
* Fog
* Rain
* Shine
* Storm
* Sunrise
* Table1 presents number of images of each category and their share in total.

**Table 1.** Categories

|  |  |  |
| --- | --- | --- |
| Class name | Images count | Percent |
| Clouds | 300 | 19% |
| Fog | 201 | 12.8% |
| Rain | 214 | 13.6% |
| Shine | 252 | 16% |
| Storm | 253 | 16% |
| Sunrise | 357 | 22.6% |
| Total | 1577 | 100% |

# Models

Three different models have been built to classify weather conditions. They are based on InceptionV3, MobileNetV2 and ResNet50 architectures and all of them were trained with the same dataset, specified in section above. As the solution uses transfer learning, models were trained on top of feature vectors provided by TensorFlow Hub [10–12]. All feature vectors were originally trained with ImageNet (ILSVRC-2012-CLS) dataset. Microsoft ML .NET library was used to train and evaluate models. The pipeline uses cross-validation with 10 numbers of folds. In machine learning cross-validation is a technique to measure the variability of a dataset. It also measures the reliability of any model trained using that data. Cross-validation algorithm divides randomly the dataset into subsets (folds). Then it builds a model on each subset and returns a set of accuracy metrics for each subset. One fold is used for validation and the others are used for training.

According to ML .NET trainer documentation [13], values of the hyperparameters are presented in Table2.

**Table 2.** Hyperparameters

|  |  |
| --- | --- |
| Hyperparameter | Value |
| Batch size | 10 |
| Epoch | 200 |
| Learning rate | 0.01 |

What are these hyperparameters?

The batch size is a number of samples processed before the model is updated.

The number of epochs is the number of complete passes through the training dataset.

Learning rate determines the step size at each iteration while moving toward a minimum of a loss function.

The source code that have been used for model creation, training and testing is publicly available on GitHub repository [14].

The most important training metrics are shown in Table3.

**Table 3.** Models evaluation metrics

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | InceptionV3 | MobileNetV2 | ResNet50 |
| Average micro-accuracy | 0.826 | 0.909 | 0.878 |
| Average macro-accuracy | 0.826 | 0.909 | 0.873 |
| Average Log-loss | 0.499 | 0.3 | 0.392 |
| Average Log-loss reduction | 0.716 | 0.829 | 0.777 |

Additionally each model pass performance and accuracy tests measuring total classification time for two groups of images of different size and aspect ratio. In this case there was no images normalization.

First experimental set contained 37 manually selected pictures, found on the web, of all six trained classes (clouds, fog, rain, shine, sunrise, storm). Among these data were many problematic items which poor quality have entailed that they were difficult to classify even for a human. The second set of data was “Multi-Class Images for Weather Classification” found on kaggle [15]. It had images of only four classes (clouds, rain, shine, sunrise), but was much larger than first experimental set - contained 1125 images. Both experimental datasets included images that were not used to create any of the models (Fig. 2).



(a) Prediction: fog, actual: rain (b) Prediction: rain, actual: rain

**Fig.2.** Example of potentially problematic images

Classifier trained on top of MobileNetV2 was almost 4 times faster than models based on the other architectures. This confirms that MobileNetV2 architecture would be the best choice for mobile and embedded systems. InceptionV3 and ResNet50 classified correctly more images than MobileNetV2. Table4 and Table5 shows time performance and accuracy for both experimental datasets.

**Table 4.** First experimental group - performance and accuracy

|  |  |  |  |
| --- | --- | --- | --- |
|  | InceptionV3 | MobileNetV2 | ResNet50 |
| Elapsed time | 95s | 25s | 99s |
| Correct classifications | 27 | 26 | 27 |
| Incorrect classifications | 10 | 11 | 10 |
| Accuracy | 0.73 | 0.70 | 0.73 |

**Table 5.** Second experimental group - performance and accuracy

|  |  |  |  |
| --- | --- | --- | --- |
|  | InceptionV3 | MobileNetV2 | ResNet50 |
| Elapsed time | 40min 03s | 10min 58s | 47min 22s |
| Correct classifications | 1075 | 1098 | 1091 |
| Incorrect classifications | 50 | 27 | 34 |
| Accuracy | 0.95 | 0.97 | 0.96 |

# Conclusion

Thanks to transfer learning it is possible to train custom classifiers without large dataset and computing power. Global efficiency and accuracy may depend on neural network architecture. All created models were able to classify weather with accuracy of 70–73% (poor quality dataset of six classes) and 95–97% (good quality dataset of four classes). InceptionV3 and ResNet50 architectures had similar classification time and accuracy. MobileNetV2 had the shortest classification time and achieved competitive results. ResNet50 model achieved a slightly higher average accuracy (based on model evaluation) than existing image-based weather classification models [1].

Selected neural networks were compared due to significant differences in architecture and the availability of vector features used for knowledge transfer. All vector features were trained on the same data set, which allowed to conclude that the differences in the achieved results result mostly from architectural differences of the compared neural networks.

Weather classification is complicated due to the difficulty of extracting the characteristics of weather phenomena. Some weather conditions are extremely difficult for classification, e.g. heavy rain may look like a fog especially if input image resolution is low.

Another difficulty is the possibility of mixing atmospheric phenomena. It happens quite often that it is rainy during storm or sun shines during rain. Such situations can not be easily resolved by simple image classification because of its limits that is necessity to select only one option (class).

An integral part of work is a source code wrote using C# programming language and ML .NET framework [14]. Despite many searches, it was not possible to find a program that would deal with similar issues, which would be built with th Attribution e mentioned technologies. Provided source code allows to follow line by line whole logic being used to build models, evaluate and use them. The use of non-mainstream technology enriches the originality of the solution.

# References

1. Xia, J., Xuan, D., Tan, L., Xing, L.: ResNet15: weather recognition on traffic roadwith deep convolutional neural network. Adv. Meteorol. **2020**, 11 (2020). Article ID 6972826
2. Grandini, M., Bagli, E., Visani, G.: Metrics for multi-class classification: anoverview (2020)
3. Zhuang, F., et al.: A Comprehensive Survey on Transfer Learning (2019)
4. An, J., Chen, Y., Shin, H.: Weather classification using convolutional neural networks, pp. 245–246 (2018). <https://doi.org/10.1109/ISOCC.2018.8649921>
5. Howard, A.G., et al.: MobileNets: Efficient Convolutional Neural Networks forMobile Vision Applications (2017)
6. Zhang, Z., Ma, H., Fu, H., Zhang, C.: Scene-free multi-class weather classificationon single images (2016)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition(2015)
8. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Inception Architecture for Computer Vision (2015)
9. ML.NET Trainer Characteristics. [https://docs.microsoft.com/en-us/dotnet/api/ microsoft.ml.vision.imageclassificationtrainer?view=ml-dotnet.](https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.vision.imageclassificationtrainer?view=ml-dotnet) Accessed 07 Nov 2021
10. Feature vector (InceptionV3). [https://tfhub.dev/google/imagenet/inception](https://tfhub.dev/google/imagenet/inception_v3/feature_vector) [v3/ feature](https://tfhub.dev/google/imagenet/inception_v3/feature_vector) [vector.](https://tfhub.dev/google/imagenet/inception_v3/feature_vector) Accessed 07 Nov 2021
11. Feature vector (MobileNetV2). [https://tfhub.dev/google/imagenet/mobilenet](https://tfhub.dev/google/imagenet/mobilenet_v2_050_224/feature_vector) [v2](https://tfhub.dev/google/imagenet/mobilenet_v2_050_224/feature_vector)

[050](https://tfhub.dev/google/imagenet/mobilenet_v2_050_224/feature_vector) [224/feature](https://tfhub.dev/google/imagenet/mobilenet_v2_050_224/feature_vector) [vector.](https://tfhub.dev/google/imagenet/mobilenet_v2_050_224/feature_vector) Accessed 07 Nov 2021

1. Feature vector (ResNet50). [https://tfhub.dev/google/imagenet/resnet](https://tfhub.dev/google/imagenet/resnet_v2_50/feature_vector) [v2](https://tfhub.dev/google/imagenet/resnet_v2_50/feature_vector) [50/ feature](https://tfhub.dev/google/imagenet/resnet_v2_50/feature_vector) [vector.](https://tfhub.dev/google/imagenet/resnet_v2_50/feature_vector) Accessed 07 Nov 2021
2. ML.NET Default Hyperparameters. [https://docs.microsoft.com/en-us/dotnet/ api/microsoft.ml.vision.imageclassificationtrainer.options?view=ml-dotnet.](https://docs.microsoft.com/en-us/dotnet/api/microsoft.ml.vision.imageclassificationtrainer.options?view=ml-dotnet) Accessed 07 Nov 2021
3. Source code. [https://github.com/pmlodzianowski/WeatherClassification.](https://github.com/pmlodzianowski/WeatherClassification) Accessed 07 Nov 2021
4. Second experimental dataset. [https://www.kaggle.com/somesh24/multiclassimages-for-weather-classification.](https://www.kaggle.com/somesh24/multiclass-images-for-weather-classification) Accessed 07 Nov 2021