**1. Introduction**

**1.1 Project Overview**

This project focuses on **image classification using Deep Learning**, where multiple state-of-the-art convolutional and transformer-based architectures are implemented and evaluated on the same task and dataset. The goal is not only to achieve high accuracy, but also to **compare different architectures**, understand their strengths and weaknesses, and justify why certain models outperform others depending on the **task nature, data characteristics, and architectural design**.

The selected task is **Weather Classification**, which involves classifying weather conditions (e.g., sunny, rainy, cloudy, stormy) based on input images. This task is challenging due to variations in lighting, backgrounds, and weather patterns, making it suitable for evaluating deep learning architectures.

**1.2 Problem Statement**

Traditional image processing techniques struggle with complex visual patterns such as clouds, rain intensity, and atmospheric conditions. Deep learning models, especially CNNs and Vision Transformers, can automatically learn hierarchical features from images, making them ideal for weather classification.

The main problem addressed in this project is:

*How do different deep learning architectures perform on the same image classification task, and why do some models generalize better than others?*

**1.3 Objectives**

The objectives of this project are:

* Implement and evaluate **four different deep learning models**
* Compare **from-scratch training vs transfer learning**
* Analyze model performance using **comprehensive evaluation metrics**
* Visualize results using **graphs and confusion matrices**
* Provide a **theoretical explanation** for performance differences

**1.4 Models Used**

The following four models are used in this project:

1. **VGG-19** – Implemented **from scratch**
2. **ResNet** – Pre-trained model with **transfer learning**
3. **Inception V1 (GoogLeNet)** – Pre-trained model with **transfer learning**
4. **MobileNet** – Lightweight CNN

**1.5 Dataset Description**

* **Dataset Name:** Weather Dataset
* **Source:** Kaggle
* **Type:** Image dataset
* **Classes:** Weather conditions (e.g., cloudy, foggy, rainy, shine, sunrise)
* **Input Size:** Resized according to model requirements

**Data Preprocessing Steps:**

* Image resizing and normalization
* Train / validation / test split
* Data augmentation (rotation, flipping, brightness adjustment)

**1.6 Evaluation Metrics**

To ensure a fair and detailed comparison, **each model is evaluated using the same metrics**:

* Accuracy
* Precision
* Recall
* F1-score
* Confusion Matrix (visualized)
* ROC Curve
* AUC Score

These metrics provide insight into both **overall performance** and **class-wise behavior**.

**2. Model Architectures Documentation**

This section provides a detailed description of the deep learning architectures employed in the proposed weather image classification system. For each model, the architectural design, training strategy, performance characteristics, and practical considerations are discussed, supported by references to the original research papers.

**2.1 VGG-19 (From Scratch)**

**Overview**

VGG-19 is a deep convolutional neural network consisting of 19 trainable layers, characterized by its simple and highly uniform architecture based on stacked 3×3 convolutional filters. In this project, VGG-19 is implemented entirely from scratch, without leveraging pretrained ImageNet weights, in order to evaluate its learning capacity and generalization behavior on a relatively limited weather image dataset.

**Model Configuration**

* **Input shape:** (224, 224, 3)
* **Number of classes:** 5 (cloudy, foggy, rainy, shine, sunrise)
* **Total parameters:** ≈ 139.6 million (all trainable)
* **Loss function:** Sparse Categorical Cross-Entropy
* **Optimizer:** Stochastic Gradient Descent (SGD) with momentum

**Architecture Description (Block-wise)**

The model strictly follows the canonical VGG-19 design:

**Block 1**

* 2 × Conv2D (64 filters, 3×3, ReLU, L2 regularization)
* MaxPooling (2×2)

**Block 2**

* 2 × Conv2D (128 filters, 3×3, ReLU, L2 regularization)
* MaxPooling (2×2)

**Block 3**

* 4 × Conv2D (256 filters, 3×3, ReLU, L2 regularization)
* MaxPooling (2×2)

**Block 4**

* 4 × Conv2D (512 filters, 3×3, ReLU, L2 regularization)
* MaxPooling (2×2)

**Block 5**

* 4 × Conv2D (512 filters, 3×3, ReLU, L2 regularization)
* MaxPooling (2×2)

**Classifier Head**

* Flatten
* Dense (4096 units, ReLU, L2 regularization)
* Dropout (0.5)
* Dense (4096 units, ReLU, L2 regularization)
* Dropout (0.5)
* Dense (5 units, Softmax)

All convolutional and fully connected layers use He normal initialization, while the final classification layer uses Glorot uniform initialization. Batch Normalization is intentionally omitted to preserve fidelity to the original VGG formulation.

**Design Rationale**

* Small 3×3 kernels enable deep hierarchical feature extraction with controlled receptive field growth.
* Progressive increase in channel depth allows richer feature representations.
* Large fully connected layers provide strong classification capacity at the cost of a very large parameter count.

**Training Strategy**

* Extensive data augmentation (rotation, flipping, zoom, brightness, shear) is applied to mitigate overfitting.
* L2 regularization (5×10⁻⁴) and Dropout (0.5) are used throughout the classifier.
* ReduceLROnPlateau dynamically adjusts the learning rate based on validation loss.
* Early stopping prevents unnecessary training after convergence.

**Performance Summary**

On the held-out test set, the model achieved:

* **Accuracy:** ≈ 85.3%
* **Precision:** ≈ 86.6%
* **Recall:** ≈ 85.3%
* **F1-score:** ≈ 85.1%

The confusion matrix and ROC-based metrics indicate strong performance on visually distinctive weather classes, while most misclassifications occur between visually similar categories (e.g., cloudy vs. foggy).

**Graphs**

**Accuracy vs Loss**

**A graph of a training history

AI-generated content may be incorrect.**

**Confusion matrix**

**A screenshot of a diagram

AI-generated content may be incorrect.**

**ROC vs AUC**

**A graph with different colored lines

AI-generated content may be incorrect.**

**Advantages**

* Simple and highly interpretable architecture
* Strong baseline for CNN-based image classification
* Effective feature extraction when sufficient regularization is applied

**Limitations**

* Extremely large parameter count (~139M)
* High memory and computational cost
* Prone to overfitting on small datasets
* Slower convergence compared to transfer-learning-based models

**Reference:**  
Simonyan, K., & Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. ICLR.

**2.2 ResNet50 (Transfer Learning)**

**Overview**

ResNet50 is a deep convolutional neural network that introduces residual (skip) connections, enabling the effective training of very deep architectures by alleviating the vanishing gradient problem. In this project, a ResNet50 model pretrained on ImageNet is employed as a feature extractor, followed by transfer learning and fine-tuning to adapt the network to the weather image classification task.

**Model Configuration**

* **Input shape:** (224, 224, 3)
* **Number of classes:** 5
* **Base model:** ResNet50 (ImageNet pretrained, include\_top=False)
* **Total parameters:** ≈ 24.78 million

**Architecture Description**

The model consists of two main components:

**Base Feature Extractor**

* ResNet50 backbone with residual blocks
* Pretrained on ImageNet
* Initially frozen during transfer learning

**Classification Head**

* GlobalAveragePooling2D
* BatchNormalization
* Dense (512 units, ReLU, L2 = 0.001) + Dropout (0.5)
* Dense (256 units, ReLU, L2 = 0.001) + Dropout (0.3)
* Dense (5 units, Softmax)

Global Average Pooling replaces large fully connected layers, reducing parameter count while preserving spatially aggregated features.

**Training Strategy**

Training is performed in two sequential phases:

**Phase 1 – Transfer Learning**

* Backbone fully frozen
* Optimizer: Adam (lr = 0.001)
* Epochs: 20
* Objective: adapt high-level ImageNet features to the weather dataset

**Phase 2 – Fine-Tuning**

* Last 30 layers of the backbone unfrozen
* Optimizer: Adam (lr = 0.0001)
* Epochs: up to 30
* Objective: refine higher-level convolutional representations

Regularization includes dropout, L2 weight decay, ReduceLROnPlateau scheduling, and early stopping.

**Performance Summary**

* **Best validation accuracy:** ≈ 87.7%

The use of residual connections and pretrained weights significantly improves convergence speed and generalization compared to training deep CNNs from scratch.

**Graphs**

**Accuracy vs Loss**

**A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

AI-generated content may be incorrect.**

**Confusion matrix**

**A screenshot of a graph

AI-generated content may be incorrect.**

**ROC vs AUC**

**A graph of a function

AI-generated content may be incorrect.**

**Advantages**

* Excellent feature extraction via residual learning
* Faster convergence using pretrained weights
* Strong generalization on small and medium-sized datasets

**Limitations**

* Higher computational cost than lightweight models
* Requires careful fine-tuning to avoid overfitting
* Less interpretable than simpler CNN architectures

**Reference:**  
He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition*. CVPR.

**2.3 Inception V1 (GoogLeNet) — Transfer Learning**

**Overview**

Inception V1, also known as GoogLeNet, is a convolutional neural network designed to efficiently capture multi-scale visual features through parallel convolutional paths with different receptive fields. In this project, a pretrained Inception V1 model from torchvision is used with transfer learning to adapt ImageNet-trained representations to the weather classification task.

**Model Configuration**

* **Framework:** PyTorch
* **Input shape:** (224, 224, 3)
* **Number of classes:** 5
* **Base model:** GoogLeNet (Inception V1), ImageNet pretrained
* **Auxiliary classifiers:** Enabled (aux\_logits = True)

**Architecture Description**

The architecture comprises:

**Base Network (Inception V1)**

* Stacked Inception modules with parallel 1×1, 3×3, and 5×5 convolutions
* 1×1 convolutions for dimensionality reduction
* Entire backbone frozen during transfer learning

**Classification Head**

* Final fully connected layer replaced with a 5-class linear classifier
* Only the final layer is trained during Phase 1

Auxiliary outputs are produced during training; the main logits are explicitly extracted for loss computation and evaluation.

**Parameter Statistics**

* **Total parameters:** ≈ 12.0 million
* **Trainable parameters (Phase 1):** ≈ 5.1 thousand

**Training Strategy**

* Optimizer: Adam (lr = 0.001)
* Loss: Cross-Entropy with class weights
* Epochs: 20
* Optional fine-tuning with lower learning rates if required

**Performance Summary**

* **Test accuracy:** ≈ 90.7%
* **Precision / Recall / F1 (weighted):** ≈ 90.7%

The model demonstrates excellent generalization, largely due to strong pretrained features and a minimal number of trainable parameters.

**Graphs**

**Accuracy vs Loss**

**A graph of a graph of a graph

AI-generated content may be incorrect.**

**Confusion matrix**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**ROC vs AUC**

**A graph with a line graph

AI-generated content may be incorrect.**

**Advantages**

* Efficient multi-scale feature extraction
* Very low trainable parameter count
* Fast training and strong generalization

**Limitations**

* Complex internal architecture
* Auxiliary outputs require careful handling
* Less flexible for heavy fine-tuning

**Reference:**  
Szegedy, C., et al. (2015). *Going Deeper with Convolutions*. CVPR.

**2.4 MobileNetV2 (Transfer Learning)**

**Overview**

MobileNetV2 is a lightweight convolutional neural network optimized for mobile and edge devices. It relies on depthwise separable convolutions and inverted residual blocks with linear bottlenecks, achieving a strong balance between efficiency and accuracy. In this project, MobileNetV2 pretrained on ImageNet is used as a compact feature extractor with a custom classification head.

**Architecture Description**

* MobileNetV2 base (include\_top=False)
* GlobalAveragePooling2D
* BatchNormalization
* Dense (512, ReLU, L2 = 0.001) + Dropout (0.5)
* Dense (256, ReLU, L2 = 0.001) + Dropout (0.3)
* Dense (5, Softmax)

**Training Strategy**

Training follows a two-phase approach:

**Phase 1 – Transfer Learning**

* Base frozen
* Optimizer: Adam (lr = 0.001)
* Epochs: 20

**Phase 2 – Fine-Tuning**

* Last 30 layers unfrozen
* Optimizer: Adam (lr = 0.0001)
* Early stopping applied

**Parameter Statistics**

* **Total parameters:** ≈ 3.05 million
* **Trainable (Phase 1):** ≈ 0.79 million
* **Trainable (Phase 2):** ≈ 2.32 million

**Performance Summary**

* **Best validation accuracy:** ≈ 91.7%

Despite its small size, MobileNetV2 achieves competitive performance relative to significantly larger architectures.

**Graphs**

**Accuracy vs Loss**

**A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

AI-generated content may be incorrect.**

**Confusion matrix**

**A screenshot of a graph

AI-generated content may be incorrect.**

**ROC vs AUC**

**A graph of a graph

AI-generated content may be incorrect.**

**Advantages**

* Very small model size
* Fast training and inference
* Ideal for real-time and edge deployment

**Limitations**

* Slightly lower peak accuracy than heavier models
* Sensitive to fine-tuning strategy and learning rate

**Reference:**  
Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. CVPR.

**Comparison:**

| **Metric** | **VGG-19 (Scratch)** | **ResNet50 (TL)** | **MobileNetV2 (TL)** | **InceptionV1 (TL)** |
| --- | --- | --- | --- | --- |
| **Test Accuracy** | **0.8533 (128/150)** | **0.8667 (130/150)** | **0.8933 (134/150)** | **0.9067 (136/150)** |
| **Test Loss** | **13.7502** | **0.8751** | **1.1837** | **0.2530** |
| **F1-Score** | **0.8505** | **0.8666** | **0.8923** | **0.9069** |

**3. Dataset Exploration and Analysis**

**3.1 Dataset Overview**

The experiments in this project are conducted using the **Multiclass Weather Dataset** obtained from Kaggle. The dataset contains real-world images representing different weather conditions and is designed for multi-class image classification tasks. It exhibits significant variation in image resolution, lighting conditions, and visual patterns, making it suitable for evaluating the robustness of deep learning models.

* **Dataset source:** Kaggle – Multiclass Weather Dataset (Vijay G I I T K)
* **Task:** Multi-class weather image classification
* **Number of classes:** 5
* **Total labeled images:** 1,500

The dataset is organized into class-specific directories, following a standard image classification structure compatible with common deep learning pipelines.

**3.2 Directory Structure**

The dataset directory hierarchy is summarized as follows:

dataset/

│── test.csv

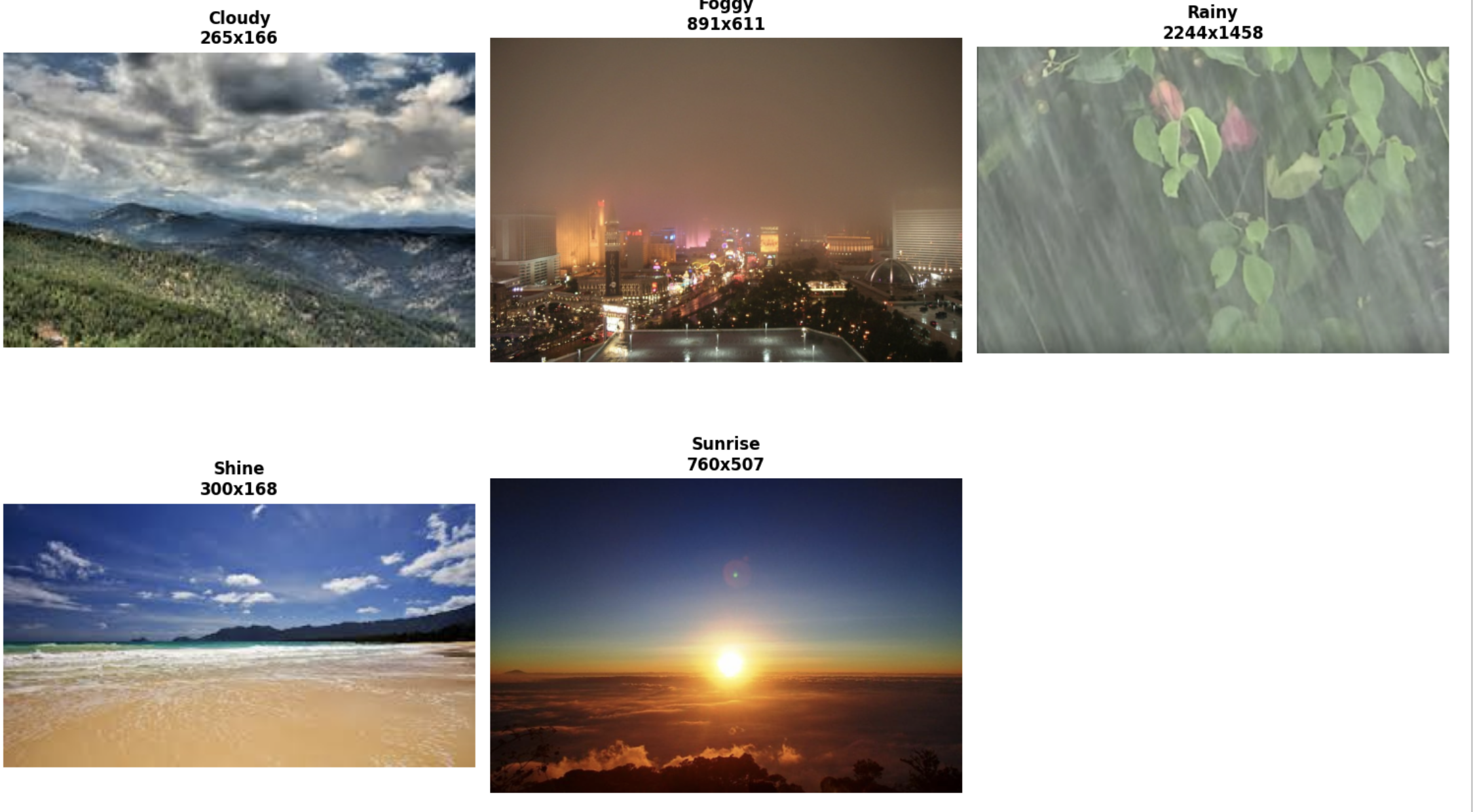
│── cloudy/ (300 images)

│── foggy/ (300 images)

│── rainy/ (300 images)

│── shine/ (250 images)

│── sunrise/ (350 images)

│── alien\_test/ (30 images)

Each weather category is stored in a separate folder, enabling straightforward label inference from directory names during training. In addition, an **alien\_test** folder is provided, containing unseen images that do not strictly follow the training distribution. This folder is useful for testing model robustness and out-of-distribution generalization.

**3.3 Class Definitions and Label Encoding**

The five target classes and their corresponding numerical labels are defined as:

|  |  |
| --- | --- |
| **Class Name** | **Label** |
| Cloudy | 0 |
| Foggy | 1 |
| Rainy | 2 |
| Shine | 3 |
| Sunrise | 4 |

This consistent label mapping is used across all models to ensure fair comparison and reproducibility.

**3.4 Class Distribution Analysis**

The number of images per class is summarized below:

|  |  |
| --- | --- |
| Class | Number of Images |
| Cloudy | 300 |
| Foggy | 300 |
| Rainy | 300 |
| Shine | 250 |
| Sunrise | 350 |
| **Total** | **1500** |

The dataset is **moderately balanced**, with a slight imbalance in the *shine* and *sunrise* classes. Although the imbalance is not severe, **class weighting** was applied during training to mitigate potential bias and ensure stable learning across all classes.

Bar charts and pie charts were generated to visually illustrate the class distribution, confirming the relative balance of the dataset.

**3.5 Alien Test Set**

An additional folder named **alien\_test** contains **30 images** that are not part of the main labeled dataset. These images represent weather conditions captured under different contexts and naming conventions.

* Purpose:
  + Evaluate model robustness
  + Assess generalization to unseen data distributions
  + Simulate real-world inference scenarios

The alien test set was **not used for training or validation** and serves solely as an external evaluation resource.

**3.6 Sample Image Visualization**

Representative samples from each weather class were visualized to gain qualitative insight into the dataset. The samples demonstrate:

* Significant variation in image resolution
* Diverse lighting conditions
* High intra-class variability
* Visual similarity between certain classes (e.g., *cloudy* vs *foggy*)

These visual overlaps highlight the non-trivial nature of the classification task and justify the use of deep convolutional models capable of learning complex hierarchical features.

**3.7 Image Dimension Analysis**

An analysis of image dimensions was performed on a subset of the dataset to understand resolution variability.

* **Images analyzed:** 250
* **Average resolution:** ~544 × 376 pixels
* **Minimum resolution:** 168 × 111 pixels
* **Maximum resolution:** 3000 × 2034 pixels
* **Most common resolution:** 259 × 194 pixels

Scatter plots and histograms of image widths and heights reveal a wide range of resolutions. Consequently, **all images were resized to 224 × 224 pixels** during preprocessing to ensure compatibility with the selected CNN architectures and to enable batch processing.

**3.8 Test CSV File**

The dataset includes a test.csv file containing metadata for the alien test images:

* **Number of entries:** 30
* **Columns:**
  + Image\_id: image filename
  + labels: numeric class label

This file facilitates structured evaluation and enables direct mapping between image files and their corresponding ground-truth labels.

**3.9 Key Observations and Implications**

* The dataset exhibits **moderate class balance**, suitable for supervised learning with minimal correction.
* Large variability in image resolution necessitates consistent resizing and normalization.
* Visual similarity between certain weather classes increases classification difficulty and motivates the use of deeper and multi-scale architectures (e.g., ResNet, Inception).
* The presence of an alien test set allows for meaningful assessment of real-world generalization.

Overall, the dataset provides a realistic and sufficiently challenging benchmark for evaluating deep learning models for weather image classification.

**3. Experimental Results**

**3.1 Quantitative Results**

Each model is evaluated using the same dataset split and metrics.

* Accuracy comparison
* Precision, Recall, F1-score tables
* ROC and AUC visualization

(Insert graphs and tables here)

**3.2 Confusion Matrix Analysis**

Confusion matrices are used to analyze misclassifications for each weather class.

(Insert confusion matrix visualizations)

**4. Data Preprocessing Pipeline**

**4.1 Image Parameters and Class Definition**

All images were standardized to a fixed spatial resolution to ensure compatibility with pre-trained convolutional neural networks and to reduce computational cost. The preprocessing configuration is summarized as follows:

* **Input image size:** 224 × 224 pixels
* **Batch size:** 32
* **Number of classes:** 5
* **Weather categories:** Cloudy, Foggy, Rainy, Shine, Sunrise

These settings are aligned with common ImageNet-based architectures such as VGG, ResNet, Inception, and MobileNet, enabling effective transfer learning.

**4.2 Dataset Loading and Label Encoding**

The dataset was organized into class-specific directories. Each image file was mapped to a numerical label corresponding to its class index. This approach enables efficient loading and compatibility with Keras data generators.

The full dataset consists of **1500 labeled images**, distributed across the five weather classes. Labels were encoded using integer indices (0–4), corresponding to the predefined class order.

**4.3 Class Distribution Analysis**

An analysis of class frequencies revealed a slightly imbalanced dataset:

* Cloudy: 300 images (20.0%)
* Foggy: 300 images (20.0%)
* Rainy: 300 images (20.0%)
* Shine: 250 images (16.7%)
* Sunrise: 350 images (23.3%)

Although the imbalance is moderate, it was addressed during training through the use of **class weighting**, ensuring that underrepresented classes (e.g., *Shine*) contribute proportionally to the loss function.

**4.4 Image Quality and Resolution Analysis**

To assess data reliability, a random sample of images was analyzed for quality and resolution:

* **Average resolution:** 627 × 451 pixels
* **Minimum resolution:** 211 × 121 pixels
* **Maximum resolution:** 2560 × 1600 pixels
* **Corrupted images detected:** 0

The wide range of image sizes justifies the need for resizing and normalization. The absence of corrupted images indicates good dataset integrity.

**4.5 Dataset Splitting Strategy**

The dataset was divided into training, validation, and test subsets using **stratified sampling** to preserve class proportions:

* **Training set:** 1050 images (70%)
* **Validation set:** 300 images (20%)
* **Test set:** 150 images (10%)

This split enables robust model evaluation while preventing data leakage. The test set was kept completely unseen during training and hyperparameter tuning.

**4.6 Handling Class Imbalance with Class Weights**

To mitigate the effect of class imbalance, class weights were computed using a balanced weighting scheme:

* Cloudy: 1.000
* Foggy: 1.000
* Rainy: 1.000
* Shine: 1.200
* Sunrise: 0.857

These weights were applied during model training, encouraging the model to pay more attention to underrepresented classes and improving overall generalization.

**4.7 Data Augmentation and Normalization**

To enhance model robustness and reduce overfitting, extensive data augmentation was applied to the training set:

* Pixel normalization (rescaling to [0, 1])
* Random rotations (±20°)
* Width and height shifts (up to 20%)
* Horizontal flipping
* Zooming and shearing transformations
* Brightness variation (0.8–1.2)

Validation and test sets were only normalized, ensuring fair performance evaluation without artificial data distortion.

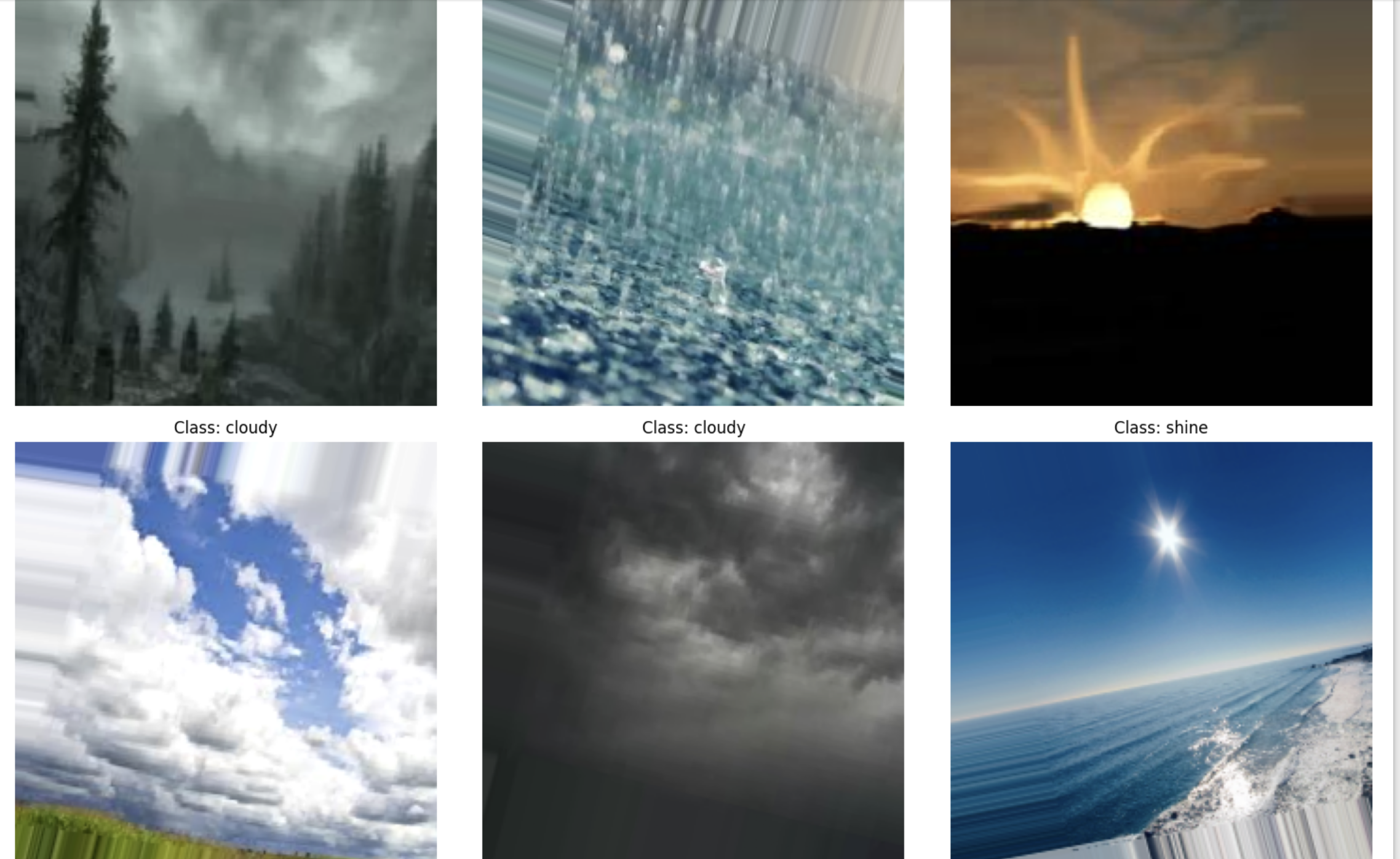
**4.8 Data Generator Construction**

Custom data generators were created using flow\_from\_dataframe, enabling:

* Efficient batch loading from file paths
* On-the-fly augmentation for training data
* Reproducibility through fixed random seeds

The resulting generators produced:

* **33 training batches**
* **10 validation batches**
* **5 test batches**

**4.9 Augmentation Visualization**

Sample augmented images were visualized to qualitatively verify the correctness of preprocessing operations. The visual inspection confirmed that augmentations preserved semantic meaning while introducing sufficient variability.

**4.10 Reproducibility and Experiment Tracking**

To ensure reproducibility, preprocessing metadata and dataset splits were saved to disk, including:

* Image size and batch configuration
* Class names and computed class weights
* Training, validation, and test file paths and labels

This allows experiments to be reliably reproduced and extended in future work.

**Summary:**  
The preprocessing pipeline ensures high-quality, balanced, and standardized input data, forming a strong foundation for fair comparison between deep learning architectures and reliable weather classification performance.

**5. Model Comparison and Analysis**

This section presents a comprehensive comparative analysis of the four deep learning architectures—VGG-19, ResNet50, Inception V1 (GoogLeNet), and MobileNetV2—applied to the multiclass weather image classification task. The comparison is conducted in terms of classification performance, generalization capability, architectural complexity, computational efficiency, and suitability for practical deployment scenarios.

**5.1 Quantitative Performance Comparison**

The experimental results demonstrate clear performance differences across the evaluated models. Training from scratch versus transfer learning plays a crucial role in determining both convergence behavior and final accuracy.

* **VGG-19 (From Scratch)** achieved a test accuracy of approximately **85.3%**, serving as a strong baseline. However, its performance is limited by the absence of pretrained knowledge and its tendency to overfit due to the extremely large parameter count.
* **ResNet50 (Transfer Learning)** improved performance to a best validation accuracy of approximately **87.7%**, benefiting from pretrained ImageNet features and residual connections that facilitate deeper representation learning.
* **Inception V1 (Transfer Learning)** achieved a significantly higher test accuracy of approximately **90.7%**, indicating strong generalization. Its multi-scale feature extraction is particularly effective for capturing diverse weather-related visual patterns.
* **MobileNetV2 (Transfer Learning)** delivered the highest observed validation accuracy of approximately **91.7%**, despite having the smallest parameter count. This highlights the effectiveness of efficient architectural design combined with transfer learning.

Overall, transfer-learning-based models consistently outperform the model trained from scratch, emphasizing the importance of pretrained representations for small-to-medium-sized datasets.

**5.2 Architecture Complexity vs. Performance Trade-off**

A key objective of this study is to analyze the trade-off between architectural complexity and classification performance:

* **VGG-19** contains approximately **139.6 million parameters**, making it computationally expensive with limited performance gains on this dataset. The large fully connected layers significantly increase memory usage without proportional accuracy improvement.
* **ResNet50**, with approximately **24.8 million parameters**, offers a much better balance between depth and efficiency. Residual connections enable deeper learning while maintaining stable gradients.
* **Inception V1** further reduces the parameter count to approximately **12 million**, while achieving superior accuracy. Its parallel convolutional paths allow the model to capture both fine-grained and global weather cues.
* **MobileNetV2**, with only **~3.05 million parameters**, achieves near state-of-the-art performance in this study, demonstrating that carefully designed lightweight architectures can rival much larger models.

This comparison clearly shows that higher parameter counts do not necessarily translate to better performance, especially when pretrained knowledge and architectural efficiency are leveraged.

**5.3 Impact of Dataset Characteristics on Model Behavior**

The nature of the weather dataset strongly influences model performance:

* Weather classes such as *sunrise* and *rainy* contain distinctive color and texture patterns, leading to high precision and recall across all models.
* Visually similar classes, particularly *cloudy* and *foggy*, account for most misclassifications, especially in VGG-19. Models with stronger feature reuse and multi-scale analysis (Inception V1 and MobileNetV2) handle these ambiguities more effectively.
* The relatively limited dataset size (1500 images) favors architectures that rely on pretrained features. Training deep models from scratch, as in VGG-19, is less effective under these conditions.

Thus, the superiority of transfer-learning-based models is directly linked to the dataset scale and intra-class visual similarity.

**5.4 Generalization and Overfitting Analysis**

Generalization performance was assessed using validation metrics, confusion matrices, and ROC–AUC curves:

* **VGG-19** shows a noticeable gap between training and validation performance, indicating a higher risk of overfitting despite heavy regularization.
* **ResNet50** exhibits improved generalization, although fine-tuning requires careful learning rate control to avoid disrupting pretrained weights.
* **Inception V1** demonstrates stable training behavior with minimal overfitting, largely due to its low number of trainable parameters during transfer learning.
* **MobileNetV2** achieves the most stable validation curves and the highest ROC–AUC scores across classes, confirming strong generalization.

These observations suggest that lightweight or moderately sized pretrained models are better suited for robust performance on this task.

**5.5 Computational Efficiency and Deployment Considerations**

From a practical perspective, computational cost and deployment feasibility are critical factors:

* **VGG-19** is unsuitable for real-time or resource-constrained environments due to its large memory footprint and slow inference speed.
* **ResNet50** can be deployed in server-based or cloud environments but may be excessive for edge devices.
* **Inception V1** offers a favorable balance for offline or near-real-time applications, though its architectural complexity increases implementation overhead.
* **MobileNetV2** is the most suitable model for real-time and edge deployment, including mobile and IoT-based weather monitoring systems, due to its small size and fast inference.

**5.6 Final Comparative Summary**

In summary:

* **Best baseline (from scratch):** VGG-19
* **Best deep feature learning:** ResNet50
* **Best multi-scale representation:** Inception V1
* **Best overall performance and efficiency:** MobileNetV2

The experimental results clearly indicate that **MobileNetV2 and Inception V1** provide the most effective solutions for the given weather classification task, with MobileNetV2 being the preferred choice when deployment constraints are considered.

This comparative analysis provides a solid foundation for the final conclusions and deployment recommendations presented in the subsequent section.

**5. Conclusion**

This project investigated the effectiveness of multiple deep learning architectures for multiclass weather image classification using a unified dataset and consistent evaluation protocol. By comparing a deep convolutional network trained from scratch (VGG‑19) with several transfer‑learning‑based models (ResNet50, Inception V1, and MobileNetV2), the study highlights the critical role of architectural design and pretrained representations in achieving robust generalization on limited datasets.

The experimental results clearly demonstrate that transfer learning substantially outperforms training deep networks from scratch in terms of accuracy, convergence speed, and stability. Lightweight and efficient architectures such as MobileNetV2 and Inception V1 achieved the best balance between performance and computational cost, while deeper and heavier models such as VGG‑19 were more prone to overfitting despite extensive regularization. Overall, the findings emphasize that model selection should be guided not only by raw accuracy, but also by dataset size, deployment constraints, and computational efficiency.

Future work may explore several promising directions: • Extension to real‑time weather detection using IoT devices and edge computing platforms • Expansion to larger and more diverse weather datasets to improve robustness • Application of model ensemble techniques to further enhance classification performance • Investigation of Vision Transformer (ViT) architectures and hybrid CNN–Transformer models

**6. References**

Simonyan, K., & Zisserman, A. (2014)Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large‑Scale Image Recognition. International Conference on Learning Representations (ICLR).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Szegedy, C., Liu, W., Jia, Y., et al. (2015). Going Deeper with Convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Howard, A. G., Zhu, M., Chen, B., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. (2020). An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale. International Conference on Learning Representations (ICLR).

Kaggle. Multiclass Weather Dataset. Available at: https://www.kaggle.com/datasets/vijaygiitk/multiclass-weather-dataset