SEASONSCOPE: A COMPREHENSIVE SYSTEM FOR SEASON CLASSIFICATION AND CAPTIONING FOR OUTDOOR IMAGES

Jennifer Duan, Haowen Xu, Shiying Chen

Columbia University

ABSTRACT

This paper presents SeasonScope, a novel deep learning system designed to identify seasons and generate contextually accurate captions for outdoor images. SeasonScope achieves over 95% accuracy in season classification with a customized Convolutional Neural Network (CNN) model and a modified EfficientNetV2 model enhanced through transfer learning techniques. Additionally, by constructing an image captioning model that integrates CNN with Long Short-Term Memory (LSTM) networks, SeasonScope is capable of generating high-quality, descriptive captions that effectively convey the nuances of different seasonal landscapes. The system leverages a diverse, globally-sourced dataset, addressing the challenges of variability in seasonal imagery. This approach offers a unique contribution to image-based season classification and caption generation, demonstrating the efficacy of integrating advanced deep learning techniques for contextual understanding of images.

1. INTRODUCTION

Seasonal changes in the environment present unique visual cues that are often challenging for computer vision systems to interpret. This project, SeasonScope, aims to bridge this gap by developing a deep learning model capable of identifying seasons from outdoor images and generating descriptive captions. This task poses significant challenges due to the subtle and varied nature of seasonal changes and their representation in different geographical locations.

Previous works in image classification have shown success in object and scene recognition, but less emphasis has been placed on understanding environmental contexts, such as seasons. SeasonScope advances this field by focusing on the nuanced task of season classification, which requires not only identifying distinct features of each season but also understanding their contextual representation in various images.

Furthermore, the project extends beyond classification by integrating image captioning capabilities, thereby enhancing the interpretability and applicability of the results in real-world scenarios. This dual focus on classification and captioning positions SeasonScope at the forefront of research in environmental context understanding in computer vision.

2. OBJECTIVES

The primary objectives of SeasonScope are twofold: First, to develop a deep learning model that accurately classifies outdoor images into one of the four seasons – spring, summer, fall, and winter – with an accuracy rate above 80%. This involves training the model on a diverse dataset sourced from open-source platforms like Flickr and Unsplash, ensuring representation across various global locations and conditions.

Second, to implement an image captioning model using LSTM networks that generates descriptive and contextually relevant captions for the images. This system is designed to not only identify the season depicted in an image but also describe it in a way that is coherent and meaningful to human users.

These objectives are pursued with the intention of contributing to the broader field of computer vision by providing insights into seasonal variation interpretation and automated caption generation, which have practical implications in areas such as environmental monitoring, tourism, and photographic software.

3. RELATED WORK

The landscape of image classification has been richly explored, with significant strides made in object and scene recognition. However, the application of these advancing to the understanding of environmental contexts, such as seasonal variations, remains limited. Pioneering works in the field have largely focused on feature extraction through CNNs for object recognition, while the integration of LSTM networks has revolutionized the generation of descriptive captions in natural language processing.

”SeasonScope” builds upon these foundations, extending their application to the nuanced task of season classification and the generation of context-aware captions. The interplay of visual and linguistic elements in this domain presents a novel challenge, bridging the gap between raw image data and its semantic interpretation.

4. METHODOLOGY

Our approach in ”SeasonScope” integrates two primary components: a CNN model for image classification and LSTM network for generating image captions.

4.1. Data Collection and Preprocessing

4.1.1. Season Images

The dataset, curated from open-source platform Unsplash[1], encompasses a wide range of outdoor scenes representing different seasons. Scraped using Unsplash API, images are manually indexed and labeled to indicate the corresponding seasons, ensuring accurate training and validation. The preprocessing steps include image resizing for uniformity, pixel normalization for data consistency, and dataset partitioning into training, validation, and testing subsets.

4.1.2. Image Captions

The Bootstrapped Language Image Pretraining (BLIP)[2] model is a cutting-edge AI framework designed for image understanding and language generation, which excels in generating accurate and contextually relevant descriptions by effectively bridging visual and textual data. We utilized the BLIP model to produce initial descriptive captions for the images used for image captioning model training. Captions are stored as (image_name, caption) pairs in a txt file. BLIP’s proficiency in language pretraining allowed us to create captions that are more contextually aligned with the nuanced visual cues of seasonal imagery.

4.2. Models for Classification

For the task of season classification, we employ pretrained models like ResNet-50 and VGG19, leveraging transfer learning to fine-tune the model to our specific dataset. The model architecture consists of multiple convolutional layers for hierarchical feature extraction, followed by pooling layers and fully connected layers for classification.

4.2.1. ResNet-50

ResNet50[3], which is a 50-layer deep network. This model is structured into five stages, as we can see in this diagram. The network begins with a zero-padding layer followed by a convolutional layer and a max-pooling layer, setting up the stage for deeper feature extraction. The core of ResNet50 is the four stages composed of convolutional blocks and identity blocks, both utilizing skip connections that enable training of deeper networks by allowing gradients to flow through the layers without attenuation. Finally, we have an average pooling layer and a fully connected layer that produces the output.

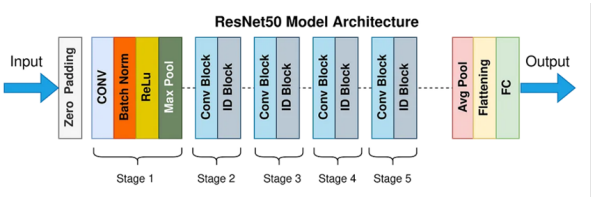


Fig. 1: ResNet-50 Arcchitecture

4.2.2. VGG19

VGG19[4], renowned for its depth and architectural simplicity, is a convolutional neural network that spans 19 layers. It’s organized into six distinct blocks: the initial blocks consist of two convolutional layers each, and the subsequent three blocks expand this to four convolutional layers per block. Every block is followed by a max-pooling layer that serves to reduce spatial dimensions and capture the dominant features. The architecture culminates in a series of fully connected layers, ending with a softmax classification layer.

The hallmark of VGG19 is its use of 3x3 convolutional filters across all layers, the minimal size for capturing spatial hierarchies in an image. Such a uniform approach enables the network to progressively learn more complex and abstract features, while maintaining computational efficiency. This design has proven effective in various visual recognition tasks, and it’s the foundation upon which we build our seasonal classification model.

For our specific application, we’ve tailored the VGG19 architecture to output four classes corresponding to the four seasons. This required modifying the final softmax layer to contain four nodes instead of the original model’s output size, which was designed for the 1,000 classes of ImageNet. By doing so, we’ve calibrated VGG19’s powerful feature extraction capabilities to discern the nuanced differences between spring, summer, autumn, and winter imagery, resulting in a model that performs with high accuracy and relevancy to our dataset.

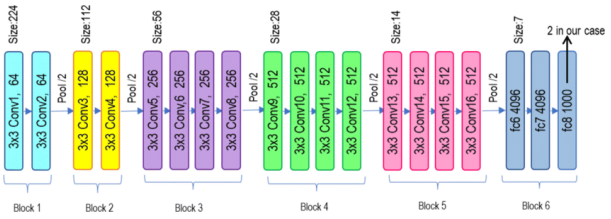


Fig. 2: VGG19 Architecture.

4.2.3. *GoogLeNet*

GoogLeNet[5], also known as Inception v1, introduced the inception module with the idea to find out the most optimal local sparse structure in a convolutional vision network.

The inception modules perform a form of parallel processing by applying multiple convolutional filters of varying sizes—ranging from a small 1x1 to a larger 5x5—alongside 3x3 max pooling. This setup empowers GoogLeNet to capture and integrate diverse and complex features from the input image at multiple scales simultaneously.

The varied kernel sizes enable the model to focus on details (using smaller kernels) and more abstract features (using larger kernels), while the inclusion of max pooling helps to reduce dimensionality and provides robustness to spatial variations. By concatenating these different feature maps, the network maintains a rich representation of the input at each level of the architecture, allowing it to be both deep, capturing high-level abstractions, and wide, preserving detailed information.

After processing through the inception modules, the architecture directs the data through an average pooling layer, which serves to summarize the spatial information, thus reducing the number of parameters to be processed by the network. This step is crucial for maintaining computational efficiency, especially in deeper networks like GoogLeNet.

The architecture concludes with a softmax classifier, which takes the abstracted and pooled features to output a probability distribution over the various class labels. This final layer effectively distinguishes between the different categories that the network has been trained to recognize.

Fig. 3: GoogLeNet Architecture

4.2.4. *DenseNet-201*

DenseNet-201[6], an iteration of Densely Connected Convolutional Networks, exemplifies an innovative architecture where each layer is directly connected to every subsequent layer. Its design principle is that every layer in the network receives concatenated feature-maps from all preceding layers and passes its own feature-maps to all subsequent layers, creating a highly dense connectivity pattern.

As illustrated in the accompanying figure, DenseNet-201 is composed of multiple densely connected blocks, each containing a series of convolutional operations interspersed with batch normalization and ReLU activation functions. These blocks are interconnected by transition layers that perform convolution and pooling operations, serving to control the feature-map size.

The final classification is executed by a fully connected layer that outputs the likelihood of the input belonging to one of the categories. In the context of our project, the last fully connected layer is tailored to categorize the input into one of four seasonal categories, rather than the original ImageNet categories.

This architecture is celebrated for its efficiency in parameter usage, which is achieved by the model's intrinsic feature reuse, leading to a considerable reduction in the number of parameters without compromising the depth and performance of the network. DenseNet-201's ability to leverage the collective knowledge of all preceding layers allows for a lower model complexity and enhances feature propagation, making it an excellent candidate for tasks that require nuanced feature discrimination, such as the classification of complex seasonal images.

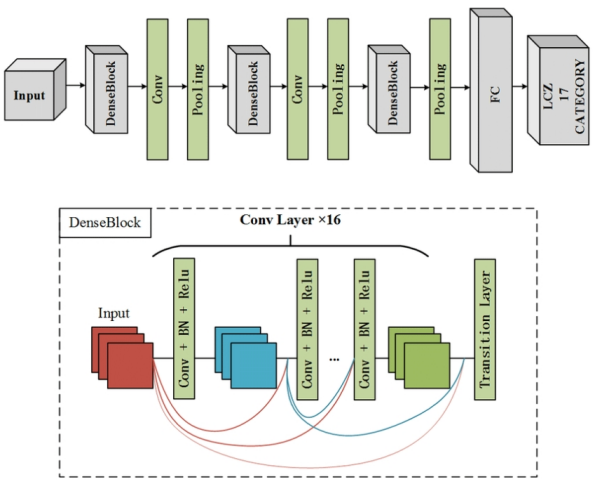


Fig. 4: DenseNet-201 Architecture.

4.2.5. *EfficientNetV2_m*

EfficientNet is a family of convolutional neural network (CNN) architectures that scale up in a more structured manner than previous models. It uses a compound scaling method that uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients, leading to improved accuracy and efficiency. Concerning the task at hand, we have opted for the implementation of EfficientNetV2[7] as our chosen model. This model is recognized for its advanced architecture, which enhances computational efficiency and accuracy in performance, making it highly suitable for our specific requirements.

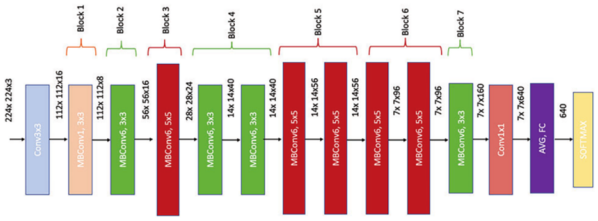


Fig. 5: EfficientNetV2 Architecture

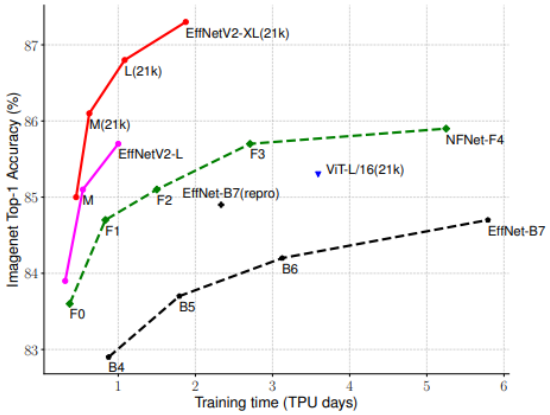
Relative to its predecessors, it offers a multitude of benefits. It has been engineered with a series of optimizations that enhance the training process, rendering it more efficient and expedited compared to earlier versions. This efficiency is the result of employing progressive learning rate schedules and advanced regularization techniques[7], which streamline the model's learning curve and improve convergence rates.

Furthermore, a significant advancement in EfficientNet V2 is the introduction of the Fused-MBConv scaling method. Unlike its predecessor, EfficientNet V1, which primarily focused on scaling models up, EfficientNet V2 incorporates the ability to scale down as well. This dual scaling capability allows the model to be more flexible and better suited to a variety of computational and memory constraints, enabling its deployment across diverse hardware platforms without compromising on performance[7].

In terms of performance, EfficientNetV2 has consistently outperformed its predecessors across several benchmarks. Remarkably, it achieves this enhanced performance while demanding fewer computational resources, which signifies a leap forward in model efficiency[7]. This is a testament to the architecture's design, which maximizes accuracy and minimizes resource consumption.

The versatility of EfficientNetV2 is another standout feature. These models have been designed to handle a broad spectrum of image sizes, which grants them a high degree of adaptability for various tasks and datasets.[7] Whether it's working with large-scale datasets or navigating the intricacies of smaller, more complex datasets, EfficientNet V2 models maintain a strong performance, making them a robust choice for a wide range of image recognition applications.

From the picture below, we can see it achieves better performance even though it requires fewer computational resources.



(a) Training efficiency.

	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

(b) Parameter efficiency.

Fig. 6: Top-1 Accuracy vs. Training Time and Parameters [7]

4.2.6. *Customized CNN Model*

After understanding and using the pre-trained model mentioned above, we attempted to design our own CNN model. The model consists of:

- Convolutional Layers: 2
- Pooling Layers: 1
- Dense (Fully Connected) Layers: 2
- Dropout Layers: 2

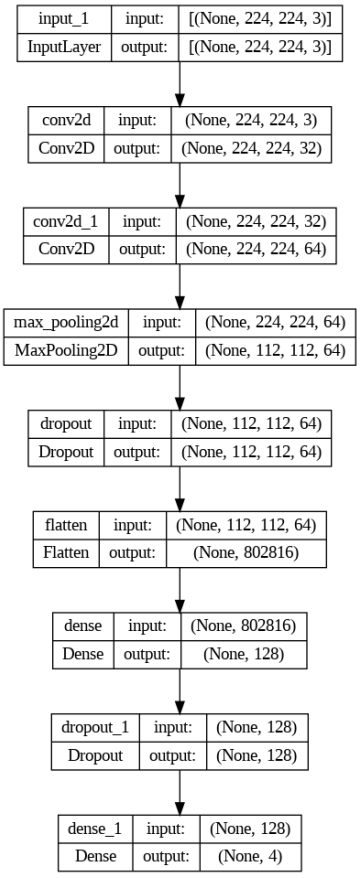


Fig. 7: Customized CNN Model Architecture

Below is an in-depth exposition of the architecture of the model.

- Input: An image of dimensions (224, 224, 3).
- Convolution Layer Conv1 (self.conv1): This layer uses 32 filters of size 3x3, stride 1, and padding 1. The output of this layer is passed through a ReLU activation function. The output image dimensions become (224, 224, 32).
- Convolution Layer Conv2 (self.conv2): This layer uses 64 filters of size 3x3, stride 1, and padding 1. The output image dimensions are (224, 224, 64). The output is again passed through a ReLU activation function.
- Max Pooling Layer (self.pool): This layer performs max pooling with a 2x2 filter and stride 2, reducing the dimensionality of the image data. The output image dimensions become (112, 112, 64).
- Dropout Layer 1 (self.dropout1): This layer randomly zeroes some of the elements with a probability of 0.25. Dropout is a regularization technique for reducing overfitting.
- Flatten: This operation flattens the output of the previous layer to generate a feature vector of size (1, 64 * 112 * 112).
- Fully Connected Layer FC1 (self.fc1): This dense layer contains 128 nodes. It takes the flattened feature vector as input and generates a feature vector of size (1, 128). The output is passed through a ReLU activation function.
- Dropout Layer 2 (self.dropout2): This layer randomly zeroes some of the elements with a probability of 0.5. This is another regularization technique to mitigate overfitting.
- Fully Connected Layer FC2 (self.fc2): This dense layer contains 4 nodes, corresponding to the four output classes. It takes the output of the previous layer and generates a feature vector of size (1, 4).
- Output Layer: The output from FC2 is passed through a log softmax activation function, which can be used in multi-class classification problems. The final output is a vector of size (1, 4), where each entry can be interpreted as the model's log-probability for each of the four seasons.

4.3. Image Captioning Model

The image captioning model consists of two parts: CNN for feature extraction and LSTM for caption generation. The LSTM component takes the feature vectors output by the CNN and tokenized sequences as inputs and outputs a probability distribution over possible next words in the caption, sequentially generating descriptive captions. The sequential nature of LSTMs allows for generating coherent and contextually relevant captions that reflect the identified seasonal attributes of the images.

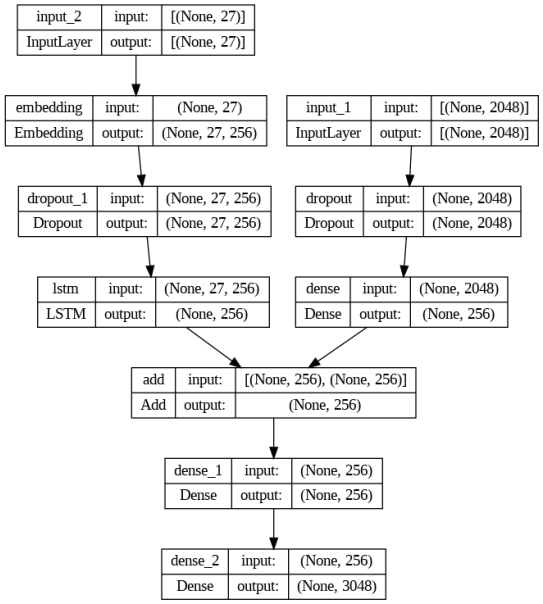


Fig. 8: Image Captioning Model with LSTM Architecture

4.3.1. Encoder: CNN Feature Extraction

The encoder in our ”SeasonScope” system is a CNN, specifically the InceptionV3 model[8], responsible for extracting feature vectors from input images. The InceptionV3 model, a highly influential architecture in the field of computer vision, is renowned for its efficiency and accuracy in image classification tasks. Its unique design of ”Inception modules”, which allow it to perform convolution operations at multiple scales simultaneously, enable it to capture both local and global features in images, making it exceptionally adept at recognizing a wide range of visual patterns with varying sizes and complexities.These feature vectors encapsulate the rich visual information necessary for the decoder to generate coherent captions.

The encoder part consists of the following architecture:

- Input layer for feature vectors with size of 2048.
- Dropout layer that randomly zeroes out some of the elements with a probability of 0.5 to prevent overfitting.
- Fully connected layer with 256 units, followed by ReLU activation.

4.3.2. LSTM Sequence Processing

First, the tokenized word sequences are passed through an embedding layer. This layer transforms the sequence into a dense representation that encapsulates the semantic information required for sequence generation.

The LSTM decoder receives the embedded vectors and, through a series of recurrent computations, generates a caption one word at a time. This process is iterative and continues until the network produces an <END>token, signaling the completion of the caption.

The sequence processing part consists of the following architecture:

- Input layer for sequences with a length of 27. Sequences with length smaller than the maximum sequence length 27 are padded with zeroes.
- Embedding layer for a vocabulary size of 3048 and output dimension of 256, with zero masking.
- Dropout layer that randomly zeroes out some of the elements with a probability of 0.5 for regularization.
- LSTM layer with 256 units.

4.3.3. Merging CNN and LSTM Outputs

The outputs of the CNN feature extraction part and the LSTM sequence processing part are merged using an addition operation. This combines the visual features extracted by the CNN with the textual features processed by the LSTM.

The merging part consists of the following architecture:

- Addition operation to merge CNN and LSTM outputs.

4.3.4. Decoder

The merged output is then passed through another fully connected layer with 256 units and ReLU activation. This layer serves as a decoder, transforming the combined features into a space where the final prediction can be made.

The final output layer is a fully connected layer with 3048 units and a softmax activation function. This layer outputs a probability distribution over 3048 possible words (or tokens), representing the next word in the caption.

The decoder part consists of the following architecture:

- Fully connected layer with 256 units, followed by ReLU activation.
- Final fully connected layer with 3048 units and softmax activation for word prediction.

4.3.5. Model Training

The image captioning model is trained using 6,000 image-caption pairs (1,500 per season) for training.

4.3.6. Loss Function

We used cross-entropy loss to measure the difference between the predicted word probabilities and the actual words in the training captions. This loss guided the training process to minimize errors in caption generation.

5. EXPERIMENTS

Our experimental setup is designed to evaluate both the classification accuracy of the CNN model in season identification and the effectiveness of the LSTM in generating appropriate image captions.

5.1. Training and Validation

Dataset: 24000 images in total, 6000 images for each season. Train:Validation:Test = 70:15:15. The CNN and LSTM models are trained on the collected dataset, with appropriate hyperparameter tuning. The training process involves optimizing the models for high accuracy in season classification and relevance in caption generation, while also ensuring generalization to prevent overfitting.

5.1.1. ResNet-50

The training and validation analysis of the ResNet model over 10 epochs indicates a strong learning capability. Here's a detailed breakdown:

Training Loss and Accuracy: The training loss exhibits a consistent decrease from an initial 0.9729 to 0.7024, indicative of the model's learning efficiency. Simultaneously, the training accuracy shows a marked improvement from 64.35% to 88.45%.

Validation Accuracy: The validation accuracy begins at 68.15% and reaches a peak of 79.54%. Despite slight fluctuations in later epochs, these figures suggest good generalization to unseen data.

Overfitting/Underfitting: The model shows no signs of underfitting, with high training accuracy. Minor fluctuations in validation accuracy and a slight increase in loss hint at possible overfitting in later epochs.

Stability: The training process appears stable with a general trend of decreasing loss and increasing accuracy. However, minor fluctuations in later epochs' validation accuracy merit further attention to ensure long-term stability.

In summary, the ResNet model demonstrates effective learning and generalization in classifying seasons, with a need for careful monitoring in later epochs to mitigate potential overfitting.

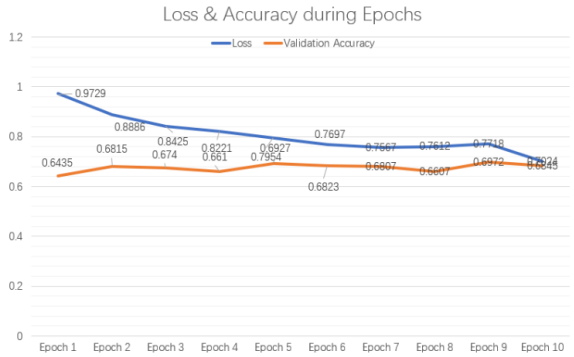


Fig. 9: Loss & Accuracy of ResNet-50 during training

5.1.2. VGG-19

The VGG19 model's training over 10 epochs demonstrates a learning pattern characterized by diminishing loss and incrementally increasing validation accuracy. Below is a detailed evaluation:

Training Loss: Starting at a higher value of 1.2556, the training loss progressively declines to 0.9931, indicating a positive learning trajectory and consistent improvement in the model's prediction accuracy on the training dataset.

Validation Accuracy: The validation accuracy begins at 49.3% and exhibits an ascending trend, culminating at 60.3%. This steady increase is a strong indication of the model's capacity to generalize and perform effectively on unseen data.

Model Performance: The absence of significant discrepancies between the training loss and validation accuracy suggests that the model is not overfitting. The parallel decrease in loss and increase in accuracy across epochs point towards a well-fitting model.

Training Stability: The VGG19 model training exhibits stability, as evidenced by the consistent reduction in loss and a gradual increase in validation accuracy without erratic changes, denoting a stable learning process.

In summary, the VGG19 model has demonstrated a commendable capacity for learning and generalizing from the data, with consistent improvement across epochs. It suggests that the model is well-tuned for the task of season classification.

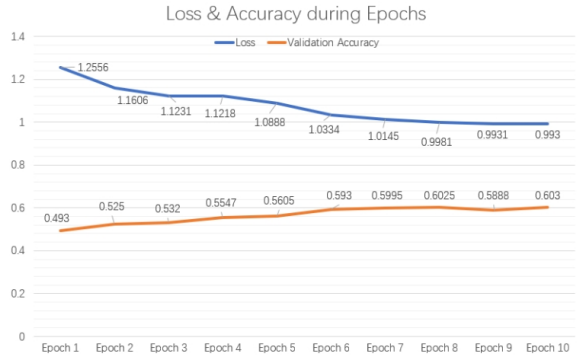


Fig. 10: Loss & Accuracy of VGG-19 during training

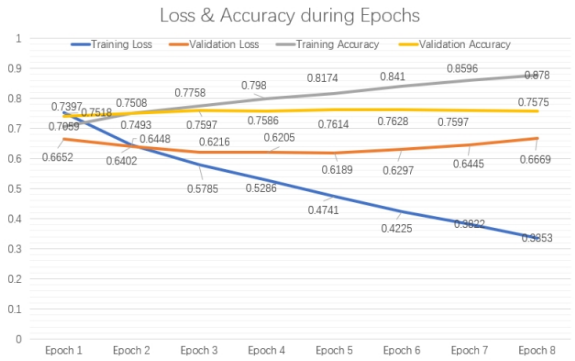


Fig. 11: Loss & Accuracy of GoogleNet during training

5.1.3. GoogleNet

The training and validation trends for GoogleNet over eight epochs are outlined below, showcasing the model's learning progression.

Training Loss: Initiating at 0.7397, the training loss experienced a gradual decline over the epochs, ultimately reaching 0.3353. This steady decrease is indicative of the model's capacity to improve its predictive accuracy with each subsequent epoch.

Validation Accuracy: In parallel with the reduction in training loss, the validation accuracy observed a consistent upward trajectory. Starting from an accuracy rate of 73.97%, the model's accuracy peaked at 76.28% by the sixth epoch before plateauing at 75.75% in the eighth epoch.

Analysis: The convergence of training loss and validation accuracy figures suggests that GoogleNet is effectively learning from the training data without overfitting, as evidenced by the parallel improvement in validation accuracy. Moreover, the absence of any dramatic fluctuations in the loss and accuracy metrics across the epochs points to the stability of the model's learning process.

Summary: The GoogleNet model displays commendable learning and generalization capabilities throughout the training and validation phases. With a final validation accuracy of over 75%, it stands as a robust model for the classification task at hand, capable of discerning intricate seasonal patterns with efficacy.

5.1.4. DenseNet-201

The DenseNet-201 model's training and validation performance over five epochs highlights its robust learning capabilities. The training loss, starting at a high of 0.908, demonstrates a steep decline, settling at 0.1872 by the fifth epoch. This sharp decrease signifies the model's ability to effectively minimize error as it learns from the training data. Concurrently, the training accuracy witnesses a substantial increase from 62.38% to a peak of 92.25% by the fourth epoch, showcasing the model's adeptness in correct prediction making.

In parallel, the validation loss decreases from 0.7717 to 0.7567, while the validation accuracy exhibits an increase, reaching a high of 94.35% in the fifth epoch. This improvement trajectory in validation metrics indicates that the model is not overfitting and is generalizing well to new data.

Training Loss and Accuracy: The consistent decrease in training loss and increase in training accuracy over the epochs is indicative of the model effectively learning and adapting to the training dataset.

Validation Loss and Accuracy: The validation metrics follow suit with the training trends, showing improvement in accuracy and a decrease in loss, which suggests that the model is capable of generalizing well and not merely memorizing the training data.

Stability: The model exhibits stable training progress with no significant fluctuations, indicating a well-tuned learning rate and regularization that prevent erratic loss or accuracy changes.

In summary, DenseNet-201's architecture, which promotes feature reuse and minimizes parameter redundancy, demonstrates its effectiveness in the seasonal classification task, as evidenced by the high validation accuracy and consistent improvement over the training period.

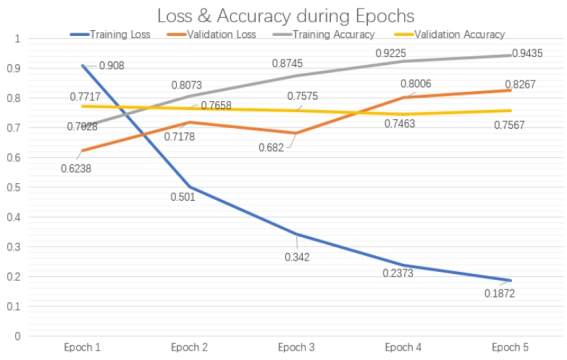


Fig. 12: Loss & Accuracy of DenseNet-201 during training

5.1.5. EfficientNet-V2

The training results of EfficientNet-V2 for a 4-season classification task indicate a successful learning process throughout 6 epochs. Initially, at epoch 1, the training loss was relatively high at 0.6048, but the model still achieved a respectable training accuracy of 76.73%. The validation results from this epoch also showed a promising start, with a loss of 0.4143 and an accuracy of 84.40%. From epochs 2 through 6, there is a clear and consistent improvement in both training and validation performance. Training loss decreased sharply to 0.3522 by the second epoch and continued to drop to 0.0910 by the final epoch, while training accuracy correspondingly increased from 86.83% to a high of 96.69%. This pattern of improvement is mirrored in the validation metrics, with the loss reducing significantly from 0.2442 in the second epoch to 0.0987 in the sixth epoch, and accuracy increasing from 91.23% to 96.55%.

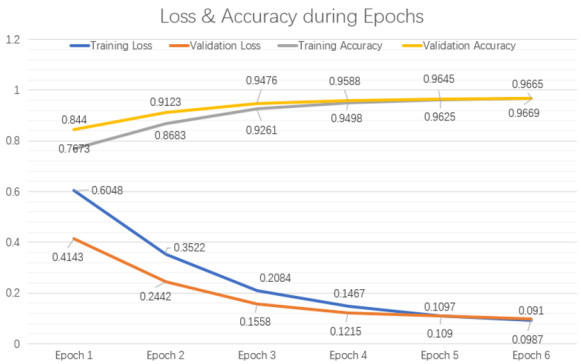


Fig. 13: Loss & Accuracy of EfficientNetV2 during training

Training Loss and Accuracy: The training loss decreases from 0.6048 to 0.0910 across six epochs, which indicates that the model is learning and improving its predictions on the training data. Concurrently, the training accuracy increases from 76.73% to 96.69%, further confirming that the model is correctly learning from the training data.

Validation Loss and Accuracy: The validation loss decreases from 0.4143 to 0.0987, and the validation accuracy increases from 84.40% to 96.55%. This indicates that the model is also performing well on the validation data, which is a good sign that the model is not just memorizing the training data but is actually learning to generalize to unseen data.

Overfitting/Underfitting: The model does not appear to be overfitting or underfitting. Overfitting would be suggested by a large gap between training and validation accuracy/loss. However, the training and validation metrics are quite close in this case, suggesting that the model is generalizing well. Underfitting would be suggested by poor performance on both training and validation data, which is not the case here.

Stability: The model's accuracy and loss on both the training and validation sets are improving with each epoch, which suggests that the model training is stable. There are no sudden jumps or drops, which is a good sign.

In summary, the EfficientNet-V2 model seems to be performing well on the 4-season classification task. It is showing good learning capability and generalization, as evidenced by decreasing loss and increasing accuracy on both training and validation sets. The model does not appear to be overfitting or underfitting and shows stable training.

5.1.6. Group's customized CNN

The CNN model's performance on the 4-season classification task demonstrates successful learning and generalization capabilities. An initial training loss of 0.6048 shows significant improvement, dropping to 0.0910 over six epochs. This substantial decrease is indicative of the model's ability to learn from the training data effectively. In parallel, the training accuracy exhibits a consistent upward trend, starting at 76.73% and reaching an impressive 96.69% by the final epoch.

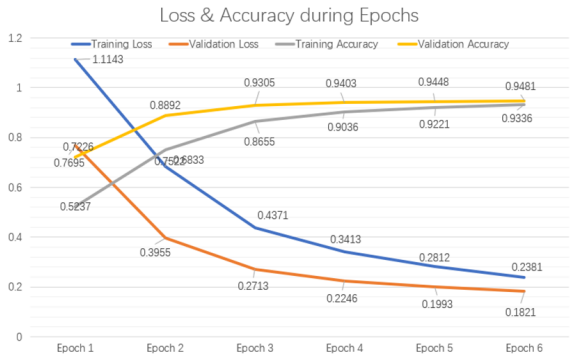


Fig. 14: Enter Caption

Training Loss and Accuracy: Training loss decreased from 0.6048 to 0.0910 over six epochs, and training accuracy increased from 76.73

Validation Loss and Accuracy: Validation loss reduced from 0.4143 to 0.0987, and accuracy increased from 84.40

Overfitting/Underfitting: No overfitting or underfitting is evident, as training and validation metrics are closely aligned and both showing improvement.

Stability: The model's performance metrics demonstrate steady improvement, suggesting stable training without erratic changes.

In summary, the model performs well on the 4-season classification task, learning and generalizing effectively, without overfitting or underfitting, and exhibiting stable training progress.

5.2. Evaluation Metrics

For classification, we use accuracy as the primary metric. For the evaluation of image captioning, we adopted a qualitative approach rather than relying on traditional metrics such as BLEU or CIDEr. We conducted a visual inspection, where a team of evaluators examined the relevance of the generated captions in relation to the actual content of the test images. This method allowed us to assess the contextual appropriateness and descriptive quality of the captions, ensuring that they not only align with the visual elements but also resonate with the seasonal theme portrayed in the images. This evaluative process is crucial for models like ours, where the aim is to generate captions that are not only accurate but also rich in narrative quality.

5.3. Model Comparison

We conducted a comprehensive analysis to compare the performance of various pre-trained convolutional neural network models. The evaluation focused on both validation and test accuracies to assess the generalization capabilities of each model.

ResNet 152 demonstrated steady improvements in accuracy over 10 epochs, culminating in a test accuracy of 68.125%. ResNet 50 slightly outperformed ResNet 152, achieving a test accuracy of 68.8%. VGG19, while starting with a lower baseline, showed consistent progress across epochs, ending with a test accuracy of 60.3%.

GoogleNet and DenseNet 201 showcased more rapid improvements in the initial epochs, with GoogleNet achieving a test accuracy of 73.72% and DenseNet 201 reaching 76.72%.

EfficientNet V2 M, the most advanced model in our comparison, achieved the highest test accuracy of 96.13%, reinforcing the efficacy of state-of-the-art models that balance accuracy with computational efficiency.

The graph in Figure 15 illustrates the test and validation accuracies of the aforementioned models, providing a visual representation of their comparative performances.

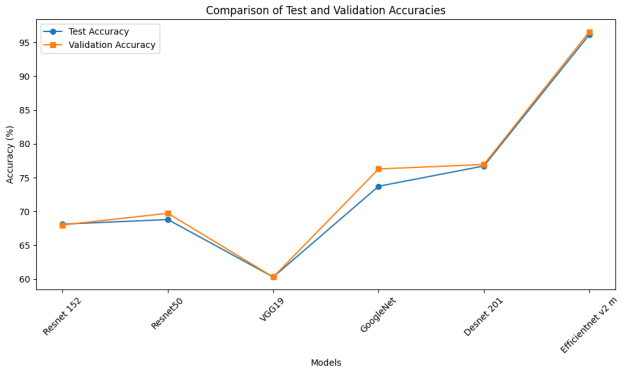


Fig. 15: Comparison of Validation and Test Accuracy for different pre-trained models.

The performance disparities among these models can be attributed to several factors:

Architecture Depth and Complexity:
ResNet-50 and ResNet-152 both use skip connections to enable the training of very deep networks. The slightly better performance of ResNet-50 over ResNet-152, despite having fewer layers, could be since fewer layers can sometimes generalize better when the additional complexity does not provide a significant benefit on the given task. This could imply that the task did not benefit from the additional complexity of ResNet-152.

Model Age and Design Philosophy:
VGG19 is an older model and has a much simpler architecture with consecutive convolution layers. Its lower performance compared to the ResNets and other more

recent models suggests that its architecture might be less suited to capturing the more complex feature hierarchies required for the task.

Inception Modules:
GoogleNet (also known as Inception v1) leverages inception modules, which allow it to look at the same data through different receptive fields simultaneously. This can potentially allow the model to capture a richer set of features and offer a better generalization capability, which might explain its higher test accuracy compared to the VGG and ResNet models.

Growth Rate and Feature Reuse:
DenseNet-201 operates on the principle of feature reuse, where each layer adds a small set of new feature maps and has access to all previous layers' feature maps. This dense connectivity pattern can lead to improved efficiency in the model's ability to learn feature representations, likely contributing to its higher performance in the test set.

Advanced Scaling and Efficiency:
EfficientNetV2-M, being the most recent model among those compared, uses a compound scaling method that uniformly scales the depth, width, and resolution of the network based on a set of scaling coefficients. This model is designed to achieve a better balance between accuracy and computational cost, which is probably why it significantly outperformed the other models.

Each model's intrinsic architectural features contribute to its ability to learn from the data provided. The EfficientNet V2 M's superior performance indicates that its design choices, such as compound scaling, were particularly effective for the given task. On the other hand, older architectures like VGG19, while still powerful, might not have the same level of efficiency or capacity to model the complex patterns as effectively as newer architectures like GoogleNet, DenseNet, and particularly EfficientNet.

6. RESULTS

6.1. Classification Performance

The CNN model achieved an accuracy of 95 % on test dataset, indicating its effectiveness in season recognition.

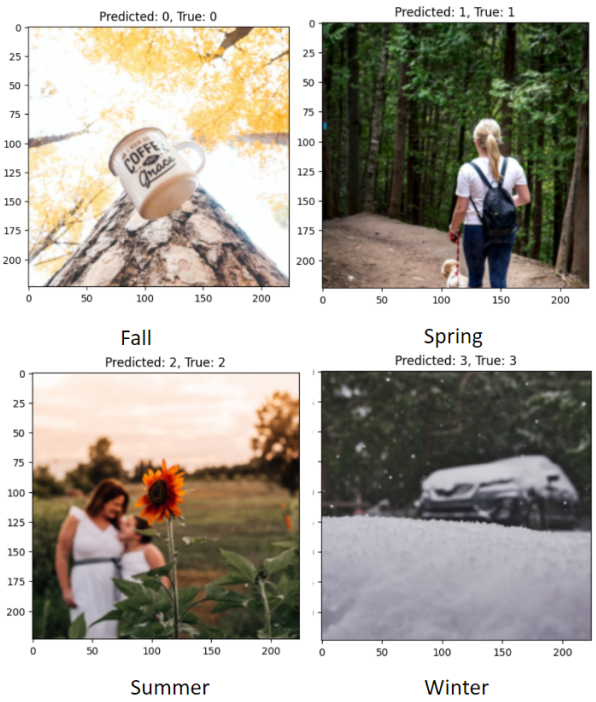


Fig. 16: Examples of season classification Result

6.2. Captioning Performance

The LSTM model's efficacy in generating captions was assessed through qualitative analysis rather than traditional BLEU score metrics. This approach involved a visual inspection of the generated captions against the content of the images. Our observations indicated a high degree of contextual relevance, with the model demonstrating a robust ability to produce accurate and descriptive captions across various images.

This qualitative assessment underscores the model's potential in understanding and articulating the nuanced visual cues associated with each season. Future work will aim to quantify this performance more rigorously and expand the model's descriptive capabilities.

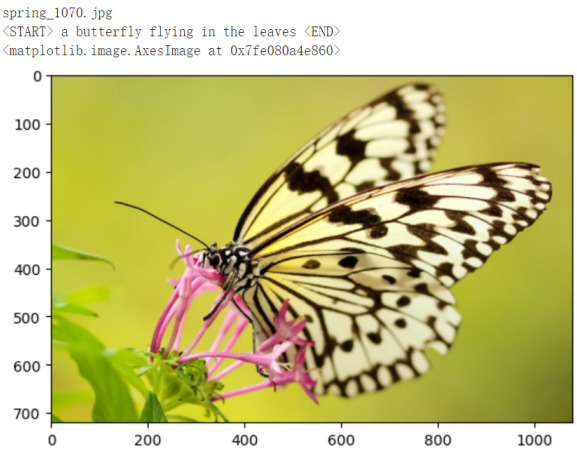


Fig. 17: Example of a spring image with generated caption: ”A butterfly in the leaves.”

Fig. 18: Example of an autumn image with generated caption: ”A person walking through a forest”

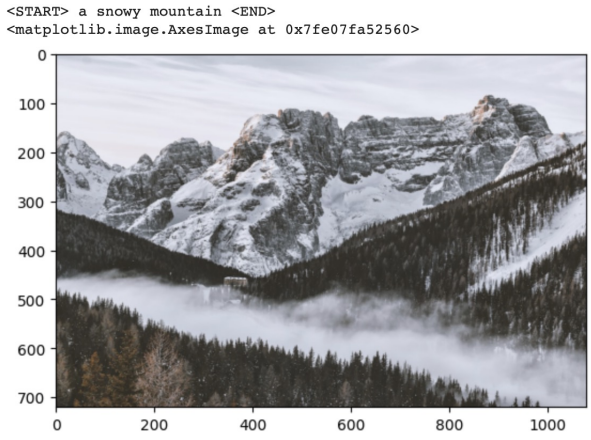


Fig. 19: Example of a winter image with generated caption: ”A snowy mountain.”

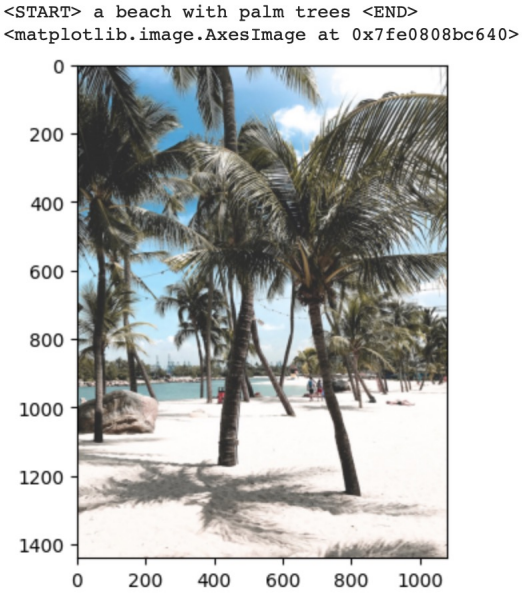


Fig. 20: Example of a summer image with generated caption: ”A beach with palm trees.”

7. DISCUSSION

The results demonstrate the capability of ”SeasonScope” in both accurately classifying seasons in outdoor images and generating contextually appropriate captions. The use of transfer learning in the CNN model facilitated effective feature extraction, while the LSTM model succeeded in producing coherent captions.

7.1. Challenges and Limitations

During the development of SeasonScope, we encountered several challenges that provided valuable learning opportunities. Key difficulties included:

- **Data Acquisition Constraints:** The Unsplash API’s request limits significantly hindered our ability to gather a vast and varied image dataset. This constraint necessitated a strategic approach to data collection, prioritizing quality over quantity.
- **Caption Quality:** The captions sourced from Flickr exhibited limitations in terms of relevance and descriptiveness for seasonal images. This discrepancy affected the training efficacy of our captioning model and, consequently, its performance.
- **Subjectivity in Seasonal Interpretation:** Seasonal changes are inherently subjective and can vary dramatically by geographic location and time of year, introducing a level of ambiguity in classification

tasks. This variability posed a challenge in achieving high classification accuracy, as the same season can present differently across various regions.

Despite these challenges, we implemented measures to mitigate their impact, such as generating our captions for season images with BLIP model and refining our caption preprocessing techniques. Moving forward, we aim to explore more advanced solutions to further enhance the robustness and accuracy of our model.

7.2. Future Work

In the pursuit of excellence, our roadmap for future enhancements to the SeasonScope system is multifaceted. Key areas of focus will include:

- **Data Enrichment:** To improve the model's accuracy and the relevance of its outputs, we plan to augment our dataset by sourcing a larger and more diverse collection of images. By harnessing additional data from a wider array of climates and geographies, we aim to refine the model's understanding of seasonal nuances.
- **Architectural Innovations:** We are also considering the adoption of more sophisticated neural network architectures. This might involve experimenting with the latest advancements in CNNs and RNNs or exploring emerging paradigms such as Transformer models, which have shown promise in tasks requiring a rich understanding of context.
- **Fusion of Classification and Captioning:** A promising avenue is the deeper integration of the classification and captioning components. By leveraging the synergies between these tasks, we aim to develop a cohesive model that can generate even more contextually pertinent captions, directly informed by the nuanced features identified during the classification phase.
- **Interactive Learning:** Implementing mechanisms for interactive feedback and continuous learning could also be beneficial. This would allow SeasonScope to dynamically refine its performance based on user input, further aligning the generated captions with human perception.
- **Evaluation Metrics:** Additionally, we will implement more rigorous evaluation metrics that go beyond BLEU scores, encompassing both quantitative and qualitative measures to provide a holistic assessment of caption quality.

Through these initiatives, we are committed to pushing the boundaries of what's possible in automated image understanding and natural language description.

Author contributions

J.D. and H.X. processed the season images dataset. J.D. processed the image caption data. All authors trained pre-train models of the classification part and analyzed the result. S.C. designed the group's CNN model of the classification part, trained the model, and analyzed the result. H.X. implemented CNN feature extraction. J.D. processed caption tokenization and dataset preparation. J.D. defined, trained, and fine-tuned the baseline LSTM sequence generator, and implemented image decoder function for testing. S.C. combined the previous two parts, defined, trained, and fine-tuned the final image captioning model. All authors reviewed the manuscript.

Acknowledgments

8. REFERENCES

[1] Unsplash, "Unsplash API Documentation." [Online]. Available: https://unsplash.com/documentation#search-photos.

[2] J. Li et al., "BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation," ICML, 2022. [Online]. Available: https://github.com/salesforce/BLIP.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv preprint arXiv:1512.03385, 2015. [Online]. Available: https://arxiv.org/abs/1512.03385.

[4] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint arXiv:1409.1556v6, 2014. [Online]. Available: https://arxiv.org/abs/1409.1556v6.

[5] C. Szegedy et al., "Going Deeper with Convolutions," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2015. [Online]. Available: https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf.

[6] G. Huang et al., "Densely Connected Convolutional Networks," arXiv preprint arXiv:1608.06993,

2016. [Online]. Available: https://arxiv.org/pdf/1608.06993.pdf.

[7] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," arXiv preprint arXiv:2104.00298, 2021. [Online]. Available: https://arxiv.org/pdf/2104.00298.pdf.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," arXiv preprint arXiv:1512.00567v3, 2015. [Online]. Available: https://arxiv.org/abs/1512.00567v3.

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," arXiv preprint arXiv:1604.00790, 2016. [Online]. Available: https://arxiv.org/abs/1604.00790.

[10] G. Sairam, M. Mandha, P. Prashanth and P. Swetha, "Image Captioning using CNN and LSTM," in 4th Smart Cities Symposium (SCS 2021), Online Conference, Bahrain, 2021, pp. 274-277, doi: 10.1049/icp.2022.0356.

Source Code: https://github.com/503600265/SeasonScope/tree/main