Image Classification with Transfer Learning

Machine Learning and Neural Networks

Meher Vinod Kumar Boggavarapu

22010815

https://github.com/bmvk1995/MyMLAssignment

Introduction:

Image classification is one of the most fundamental problems in computer vision. It refers to the task of categorizing images into predefined classes. For instance, in medical imaging, an image might be classified as "normal" or "cancerous." Similarly, in retail, product images can be grouped into categories like "electronics," "apparel," and "groceries."

However, training models for image classification from scratch poses several challenges, such as requiring large datasets, high computational resources, and extensive training time. Transfer learning has emerged as a practical and efficient solution. By leveraging pre-trained models, transfer learning enables faster development of accurate models even when labelled data is limited.

This guide demonstrates how transfer learning can be applied using VGG16, a pre-trained model, to classify images of flowers into five categories: daisy, dandelion, rose, sunflower, and tulip. It provides a step-by-step explanation, from understanding transfer learning basics to implementing and evaluating models using TensorFlow.

What is Image Classification?

Image classification is the process of identifying and assigning one or more labels to an image. Modern classification models rely on Convolutional Neural Networks (CNNs), which are well-suited for visual data due to their ability to detect patterns and features in an image.

How CNNs Work

CNNs consist of multiple layers designed to process and learn from image data:

- Convolutional Layers: These apply filters to the input image, detecting patterns such as edges, corners, and textures.
- Pooling Layers: Reduce the spatial dimensions of feature maps, making the network more efficient while retaining important information.
- Fully Connected Layers: Map the extracted features to class probabilities.

Image classification is the process of assigning one or more labels to an image based on its content. For example, an image of a dog is labelled as "dog." The task involves

analysing the visual elements (like edges, textures, and colours) to determine the image's category.

Why is Image Classification Important?

Image classification has numerous real-world applications:

- Healthcare: Classifying medical images (e.g., X-rays as "normal" or "abnormal").
- Retail: Organizing products into categories based on images.
- Security: Recognizing faces or objects in surveillance footage.

Challenges in Image Classification

- Data Requirements: Requires large labeled datasets.
- Computational Costs: Training deep learning models is resource-intensive.
- Generalization: Models need to perform well on unseen data without overfitting.

How Does Image Classification Work?

Input: An image is input to the model.

Feature Extraction: Key features like edges and shapes are extracted.

Classification: Extracted features are mapped to categories using a classifier.

Intermediate Knowledge: What is Transfer Learning?

Transfer learning is a machine learning approach where a model trained on one task is reused for another. For example, a model trained to classify general objects (e.g., cars, cats, etc.) can be repurposed to classify specific flowers (e.g., daisies, tulips).

Why Transfer Learning?

Efficiency: Reduces the time and computational power needed to train models.

Data Efficiency: Works well with smaller datasets.

Improved Accuracy: Leverages knowledge learned from large datasets like ImageNet.

Types of Transfer Learning

Feature Extraction:

Freeze pre-trained layers (convolutional layers) and replace the classifier with task-specific layers.

Example: Retain the feature-extraction layers of a pre-trained model like VGG16 and replace the top layer for flower classification.

Fine-Tuning:

Unfreeze some or all pre-trained layers and retrain them alongside the new layers.

Fine-tuning helps the model adapt to the new task while retaining general knowledge.

Pre-Trained Models for Transfer Learning

- VGG16: Known for simplicity and depth; good for feature extraction.
- ResNet50: Uses residual connections to handle deeper networks effectively.
- MobileNet: Optimized for mobile devices.
- EfficientNet: Balances depth, width, and resolution for improved performance.

Advanced Knowledge: Image Classification Using Transfer Learning

The Transfer Learning Workflow

Prepare the Dataset:

Load and preprocess images (e.g., resizing and normalization).

Split the dataset into training and validation sets.

Load the Pre-Trained Model:

Use a model like VGG16 pre-trained on ImageNet.

Exclude the top layers to use it as a feature extractor.

Add Custom Layers:

Add a global pooling layer and dense layers for classification.

Train the Model:

Start with feature extraction.

Optionally, fine-tune the model by unfreezing specific layers.

Evaluate Performance:

Use metrics like accuracy, precision, recall, and a confusion matrix.

Medical Imaging

Task: Detecting diseases in X-rays or MRIs.

Benefit: Reduces the reliance on large annotated datasets.

Retail

Task: Categorizing product images for inventory management.

Benefit: Automates cataloging processes.

Autonomous Vehicles

Task: Recognizing pedestrians and road signs.

Benefit: Ensures safety and accuracy in real-time systems.

Wildlife Conservation

Task: Identifying animal species in camera trap images.

Benefit: Facilitates biodiversity monitoring.

Challenges in Transfer Learning

Overfitting During Fine-Tuning:

Requires careful regularization and data augmentation to avoid overfitting.

Dataset Bias:

Pre-trained models may have biases due to the nature of their training data (e.g., ImageNet).

Computational Resources:

Fine-tuning large models can still be resource-intensive.

Future Research Directions

Adapting Pre-Trained Models for Video Data:

Extending transfer learning to action recognition in videos.

Efficient Transfer Learning:

Using architectures like EfficientNet to optimize performance and resource usage.

Explainability in Transfer Learning:

Developing techniques to understand why the model makes specific predictions.

Why Image Classification is Challenging

High Dimensionality: Images are large matrices of pixel values, which can be computationally expensive to process.

Data Requirements: Training a robust model requires large datasets to cover variations in lighting, angles, and backgrounds.

Overfitting: Small datasets can cause models to memorize the training data instead of generalizing to new images.

3. Understanding Transfer Learning

Transfer learning refers to reusing a pre-trained model for a different but related task. For example, a model trained to classify objects in ImageNet can be adapted to classify specific flowers. The key idea is that the early layers of a CNN capture generic patterns, such as edges and textures, which are useful across various image recognition tasks.

Types of Transfer Learning

Feature Extraction: In this approach:

The pre-trained model's convolutional layers are frozen (i.e., their weights are not updated).

Only the classifier (the top fully connected layers) is replaced and trained on the new dataset.

Fine-Tuning: This involves:

Unfreezing some or all layers of the pre-trained model.

Retraining these layers alongside the classifier to better adapt to the new dataset.

4. Applications of Transfer Learning

Transfer learning has revolutionized various fields by enabling the development of models with limited data. Below are some of its key applications:

Medical Imaging:

Task: Classifying X-rays or detecting tumors in MRIs.

Benefit: Reduces the need for large medical image datasets, which are often hard to obtain.

Autonomous Vehicles:

Task: Identifying pedestrians, road signs, or vehicles.

Benefit: Pre-trained models generalize well to diverse environments.

Retail:

Task: Grouping product images into categories.

Benefit: Simplifies catalog management for e-commerce platforms.

Security and Surveillance:

Task: Facial recognition or identifying suspicious activities.

Benefit: Enhances accuracy in systems with real-time requirements.

5. Dataset Overview

The Flower Photos Dataset is used in this project to classify images into the following five categories:

Daisy

Dandelion

Rose

Sunflower

Tulip

Dataset Characteristics

Contains a mix of close-ups and distant shots.

Features variations in lighting, angle, and background.

Simulates real-world conditions for classification tasks.

Preprocessing Steps

Resize all images to 224×224224 \times 224224×224 pixels to match the input size expected by VGG16.

Normalize pixel values to the range [0,1][0, 1][0,1] to facilitate efficient training.

6. Implementation Workflow

The transfer learning implementation follows these key steps:

6.1 Dataset Preparation

The dataset is loaded using TensorFlow's image_dataset_from_directory, split into training and validation sets, and normalized for compatibility with the pre-trained model.

python

Copy code

```
from tensorflow.keras.utils import image_dataset_from_directory
from tensorflow.keras.layers import Rescaling

# Load and preprocess dataset
data_dir = "path_to_flower_dataset"
img_height, img_width = 224, 224
batch_size = 32

train_ds = image_dataset_from_directory(
  data_dir, validation_split=0.2, subset="training",
  seed=123, image_size=(img_height, img_width), batch_size=batch_size
)

val_ds = image_dataset_from_directory(
  data_dir, validation_split=0.2, subset="validation",
  seed=123, image_size=(img_height, img_width), batch_size=batch_size
)

# Normalize images
normalization_layer = Rescaling(1./255)
train_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
val_ds = val_ds.map(lambda x, y: (normalization_layer(x), y))
```

6.2 Model Creation

We use VGG16, pre-trained on ImageNet, as the base model. The top layers are replaced with custom dense layers for the flower classification task.

python

Copy code

```
from tensorflow.keras.applications import VGG16

from tensorflow.keras import layers, models


base_model = VGG16(input_shape=(224, 224, 3), include_top=False,
weights="imagenet")

base_model.trainable = False


model = models.Sequential([

  base_model,

  layers.GlobalAveragePooling2D(),

  layers.Dense(128, activation='relu'),

  layers.Dropout(0.5),

  layers.Dense(5, activation='softmax')

])


model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

6.3 Training

The model is trained for 10 epochs, focusing on feature extraction.

python

Copy code

```
history = model.fit(train_ds, validation_data=val_ds, epochs=10)
```

6.4 Fine-Tuning

Fine-tuning is performed by unfreezing some layers of the base model to allow the network to better adapt to the flower dataset.

python

Copy code

```
base_model.trainable = True
```

```
for layer in base_model.layers[:10]:
    layer.trainable = False


model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])

fine_tune_history = model.fit(train_ds, validation_data=val_ds, epochs=5)
```

Evaluation and Results:

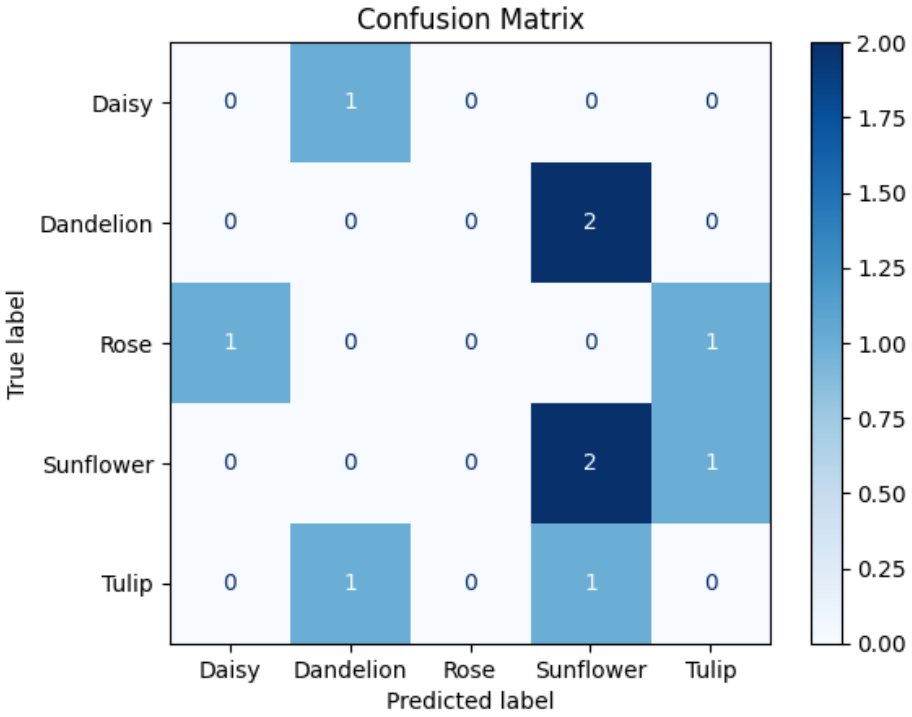
Accuracy and Loss Plots

The model's performance is evaluated using training and validation accuracy and loss curves. These plots provide insights into the learning behavior and generalization capability of the model.

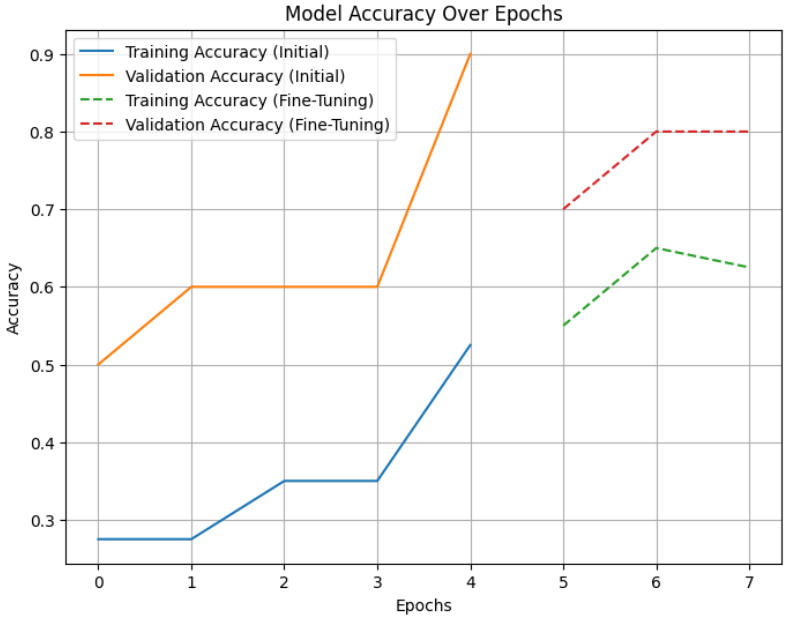
Confusion Matrix:

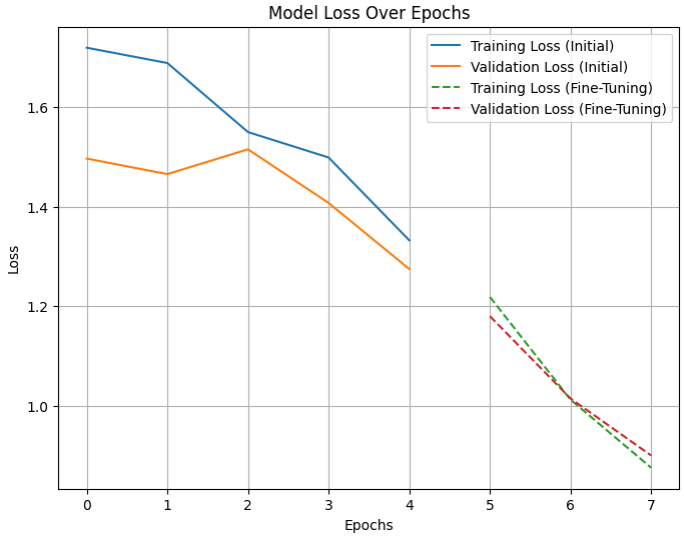
A confusion matrix visualizes how well the model performs across different classes. It highlights correct predictions and common misclassifications.

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

y_true = np.concatenate([y for x, y in val_ds], axis=0)

y_pred = np.argmax(model.predict(val_ds), axis=-1)

cm = confusion_matrix(y_true, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Daisy',
'Dandelion', 'Rose', 'Sunflower', 'Tulip'])

disp.plot(cmap='Blues')

plt.show()
```


Confusion Matrix
Daisy
Dandelion
Rose
Sunflower
Tulip
True label
Predicted label
0 1 0 0 0
0 0 0 2 0
1 0 0 0 1
0 0 0 2 1
0 1 0 1 0
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

Plots:


Model Accuracy Over Epochs
Training Accuracy (Initial)
Validation Accuracy (Initial)
Training Accuracy (Fine-Tuning)
Validation Accuracy (Fine-Tuning)
Accuracy
Epochs
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0 1 2 3 4 5 6 7



Key Insights:

- Strengths:

The model quickly achieved high accuracy by leveraging pre-trained features.

Fine-tuning further improved performance on the flower dataset.

- Weaknesses:

The model struggled with misclassifications in visually similar classes (e.g., daisy vs. dandelion).

- Future Directions:

Experiment with different pre-trained models (e.g., ResNet, EfficientNet).

Apply data augmentation techniques to increase diversity in the training dataset.

Conclusion:

Transfer learning offers a powerful solution for building image classification models efficiently. By reusing pre-trained models, we can achieve high accuracy even with limited data. This tutorial demonstrated how to implement transfer learning using VGG16 for flower classification, highlighting its strengths, limitations, and potential for future applications.

References:

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.

Chollet, F. (2017). Deep Learning with Python. Manning Publications.

TensorFlow Documentation: https://www.tensorflow.org/