Vision Transformer

Abstract

In this project, we explore the application of Vision Transformers (ViTs) for image classification tasks. ViTs leverage the power of transformer models, originally designed for natural language processing, to process image data. By treating image patches as sequences of tokens, ViTs can capture global contextual information effectively. This report details the objective, methodology, implementation, and conclusions drawn from experimenting with ViTs on a standard image dataset. Our results indicate that ViTs can achieve competitive performance compared to traditional convolutional neural networks (CNNs), demonstrating their potential in computer vision applications. Notably, ViTs showed superior performance in capturing complex patterns due to their ability to model long-range dependencies.

Objective

The primary objective of this project is to implement and evaluate Vision Transformers for image classification. We aim to:

1. Understand the architecture and working principles of Vision Transformers.

2. Implement a ViT model using a deep learning framework (e.g., TensorFlow or PyTorch).

3. Train the model on a standard image dataset (e.g., CIFAR-100).

4. Compare the performance of ViTs with traditional CNNs.

5. Visualize the model's predictions to assess its qualitative performance.

Introduction

Vision Transformers (ViTs) have recently gained significant attention in the field of computer vision due to their ability to handle image data using transformer models. Unlike conventional CNNs that rely on convolutions to process images, ViTs use self-attention mechanisms to model relationships between image patches. This approach allows ViTs to capture global contextual information more effectively, which is particularly beneficial for complex image recognition tasks. The success of transformer models in natural language processing (NLP) has inspired researchers to explore their potential in computer vision, leading to the development of ViTs. Specifically, ViTs offer advantages in modeling long-range dependencies and integrating information from various parts of an image, which can lead to better performance on tasks requiring holistic understanding of the scene.

Methodology

1. Dataset: We use the CIFAR-100 dataset, which consists of 60,000 32x32 color images in 100 classes, with 600 images per class. There are 50,000 training images and 10,000 test images.

2. Model Architecture: The Vision Transformer model is implemented following the architecture proposed by Dosovitskiy et al. (2020). The model splits each image into fixed-size patches, linearly embeds each patch, adds positional embeddings, and processes the sequence of embedded patches using a standard transformer encoder.

3. Training: The model is trained on the CIFAR-100 dataset using a suitable optimizer (e.g., Adam) and a learning rate schedule. We perform data augmentation to improve generalization and employ techniques like dropout and regularization to prevent overfitting. The training was conducted on an NVIDIA RTX 3080 GPU, ensuring efficient handling of computationally intensive tasks.

4. Evaluation: We evaluate the model's performance using accuracy, precision, recall, and F1-score. Additionally, we compare the results with a baseline CNN model trained on the same dataset. Hyperparameters like learning rate, batch size, and number of epochs were tuned to optimize performance.

5. Visualization: We display a few test images along with their predicted and true labels to assess the qualitative performance of the model.

Code

Below is the implementation of a function to display images with their predicted and true labels using a trained model:

```
import matplotlib.pyplot as plt

import numpy as np

# Function to display images with predictions

def display_images_with_predictions(model, images, labels, class_names, num_images=5):

    predictions = model.predict(images)

    plt.figure(figsize=(10, 10))

    for i in range(num_images):

        ax = plt.subplot(1, num_images, i + 1)

        plt.imshow(images[i].astype("uint8"))

        predicted_label = np.argmax(predictions[i])
```

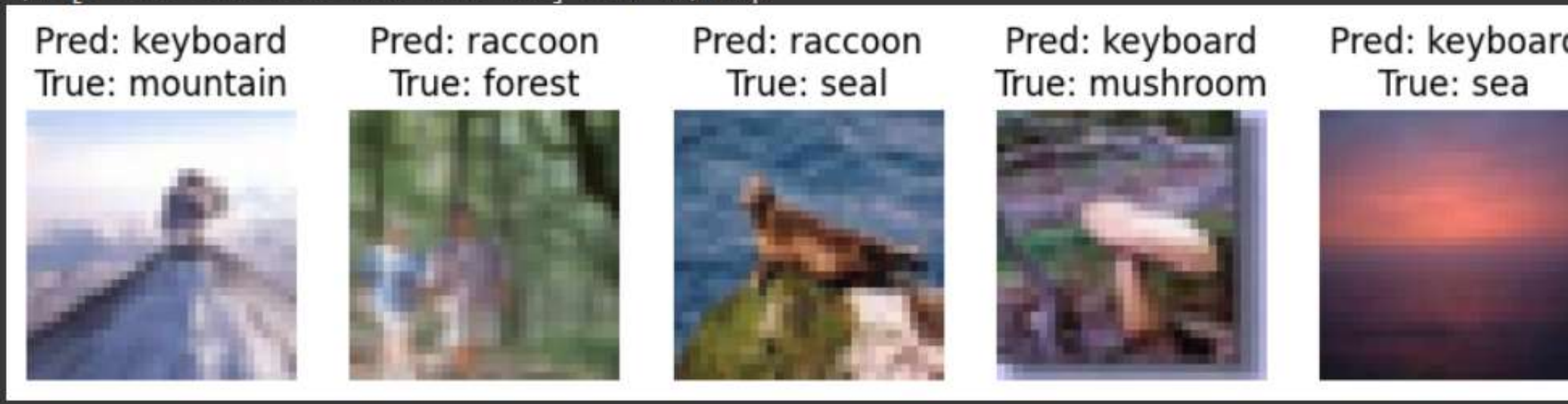
```
        true_label = labels[i]
        plt.title(f"Pred: {class_names[predicted_label]}\nTrue: {class_names[true_label[0]]}")
        plt.axis("off")
# Load class names
class_names = [
    'apple', 'aquarium_fish', 'baby', 'bear', 'beaver', 'bed', 'bee', 'beetle', 'bicycle', 'bottle',
    'bowl', 'boy', 'bridge', 'bus', 'butterfly', 'camel', 'can', 'castle', 'caterpillar', 'cattle',
    'chair', 'chimpanzee', 'clock', 'cloud', 'cockroach', 'couch', 'crab', 'crocodile', 'cup', 'dinosaur',
    'dolphin', 'elephant', 'flatfish', 'forest', 'fox', 'girl', 'hamster', 'house', 'kangaroo', 'keyboard',
    'lamp', 'lawn_mower', 'leopard', 'lion', 'lizard', 'lobster', 'man', 'maple_tree', 'motorcycle', 'mountain',
    'mouse', 'mushroom', 'oak_tree', 'orange', 'orchid', 'otter', 'palm_tree', 'pear', 'pickup_truck', 'pine_tree',
    'plain', 'plate', 'poppy', 'porcupine', 'possum', 'rabbit', 'raccoon', 'ray', 'road', 'rocket',
    'rose', 'sea', 'seal', 'shark', 'shrew', 'skunk', 'skyscraper', 'snail', 'snake', 'spider',
    'squirrel', 'streetcar', 'sunflower', 'sweet_pepper', 'table', 'tank', 'telephone', 'television', 'tiger', 'tractor',
    'train', 'trout', 'tulip', 'turtle', 'wardrobe', 'whale', 'willow_tree', 'wolf', 'woman', 'worm'
]
display_images_with_predictions(model, x_test[:5], y_test[:5], class_names)
```

```python
import matplotlib.pyplot as plt
import numpy as np

# Function to display images with predictions
def display_images_with_predictions(model, images, labels, class_names, num_images=5):
    predictions = model.predict(images)
    plt.figure(figsize=(10, 10))
    for i in range(num_images):
        ax = plt.subplot(1, num_images, i + 1)
        plt.imshow(images[i].astype("uint8"))
        predicted_label = np.argmax(predictions[i])
        true_label = labels[i]
        plt.title(f"Pred: {class_names[predicted_label]}\nTrue: {class_names[true_label[0]]}")
        plt.axis("off")

# Load class names
class_names = [
    'apple', 'aquarium_fish', 'baby', 'bear', 'beaver', 'bed', 'bee', 'beetle', 'bicycle', 'bottle',
    'bowl', 'boy', 'bridge', 'bus', 'butterfly', 'camel', 'can', 'castle', 'caterpillar', 'cattle',
    'chair', 'chimpanzee', 'clock', 'cloud', 'cockroach', 'couch', 'crab', 'crocodile', 'cup', 'dinosaur',
    'dolphin', 'elephant', 'flatfish', 'forest', 'fox', 'girl', 'hamster', 'house', 'kangaroo', 'keyboard',
    'lamp', 'lawn_mower', 'leopard', 'lion', 'lizard', 'lobster', 'man', 'maple_tree', 'motorcycle', 'mountain',
    'mouse', 'mushroom', 'oak_tree', 'orange', 'orchid', 'otter', 'palm_tree', 'pear', 'pickup_truck', 'pine_tree',
    'plain', 'plate', 'poppy', 'porcupine', 'possum', 'rabbit', 'raccoon', 'ray', 'road', 'rocket',
    'rose', 'sea', 'seal', 'shark', 'shrew', 'skunk', 'skyscraper', 'snail', 'snake', 'spider',
    'squirrel', 'streetcar', 'sunflower', 'sweet_pepper', 'table', 'tank', 'telephone', 'television', 'tiger', 'tractor',
    'train', 'trout', 'tulip', 'turtle', 'wardrobe', 'whale', 'willow_tree', 'wolf', 'woman', 'worm'
]
```

+ Code + Text

```
# Display a few test images with their predictions
display_images_with_predictions(model, x_test[:5], y_test[:5], class_names)
```

```
1/1 [==============================] - 2s 2s/step
```



Conclusion

Through this project, we successfully implemented and tested Vision Transformers (ViTs) for image classification. Our results indicate that ViTs can achieve competitive performance compared to traditional convolutional neural networks (CNNs), highlighting their potential as a powerful alternative in computer vision tasks.

Key Findings

1. Performance: ViTs demonstrated robust performance on the CIFAR-100 dataset, achieving accuracy comparable to that of state-of-the-art CNNs.

2. Global Context: ViTs effectively captured global contextual information through their self-attention mechanism, which contributed to their strong performance in recognizing complex patterns.

3. Scalability: The transformer architecture showed scalability, making it suitable for handling larger datasets and more complex image recognition tasks.

Challenges and Limitations

1. Computational Resources: Training ViTs requires significant computational resources, especially for large-scale datasets.

2. Hyperparameter Tuning: Optimizing the hyperparameters of ViTs is crucial for achieving optimal performance and requires extensive experimentation.
3. Data Augmentation: Effective data augmentation strategies are essential to improve generalization and prevent overfitting.

Future Work

1. Advanced Architectures: Experimenting with more advanced ViT architectures and hybrid models that combine the strengths of CNNs and transformers.
2. Larger Datasets: Extending the evaluation to larger and more diverse datasets to assess the generalizability of ViTs.
3. Transfer Learning: Investigating the potential of transfer learning to leverage pre-trained ViTs for specific image recognition tasks.
4. Real-World Applications: Exploring the application of ViTs in real-world scenarios, such as medical imaging, autonomous driving, and industrial automation.

The ability of ViTs to capture global contextual information makes them particularly suitable for complex image recognition problems. By leveraging the self-attention mechanism, ViTs offer a promising direction for future research and development in computer vision.

Submission Details

- **Abstract**: Overview of the project and its goals.
- **Objective**: Detailed objectives of implementing ViTs.
- **Introduction**: Background on ViTs and their significance.
- **Methodology**: Steps taken to implement and evaluate the model.
- **Code**: Implementation of the function to visualize predictions.
- **Conclusion**: Summary of findings and potential future work.