To examine the ability of Convolutional Neural Networks (CNNs) in feature detection but we will further narrow that down to edge detection across different images. It also extends to tree crown detection in aerial or satellite imagery for applications in environmental monitoring, forestry management and ecological studies.

Key goals include:

- The construction of a powerful CNN model for edge detection by tuning its architecture with regards to depth, kernel size, and activation function.
- Development and training of a method targeting tree crown detection on high resolution ortho-photo imagery, using a CNN specifically tailored for the detection, and enforcing the use of global shape, texture, color variations under a wide range of tree species and conditions.
- A dataset is curated from various sources to train and generalize the CNN models quite effectively after preprocessing it
- Training and Tuning the CNNs including transfer learning and tuning hyperparameters.
- Measuring the model performance with strict evaluation metrics using other bench mark methods and latest methodologies available.
- Modifying the outputs of models so that they can be easily interpreted and visualized in order to comprehend and trust their decision-making procedures.
- Practically illustrating how CNN-based feature detection models are used in various real life situation including medical imaging, object detection, and self-driving systems.

All together, the study is meant at advancing computer vision by demonstrating that CNNs are adaptable and effective tools for feature extraction as well as indicate their possible application areas in different subject matters.

EDGE DETECTION:

Edge detection is a technique used in computer vision and image processing to identify the boundaries or edges of objects within an image. Think of it like drawing an outline around different objects in a photo, helping to separate one object from another.

Understanding of edge detection:

1. Imagine a Photo: Take a picture with various objects in it, like a table, chair, and a vase.

2. Finding Borders: Edge detection is like finding and highlighting the borders where these objects meet the background or other objects.

3. Simplifying Details: Instead of focusing on every tiny detail, edge detection looks for places where there are sharp changes in color or brightness – these usually indicate the edge of an object.

4. Drawing Lines: Once these changes are found, edge detection can draw lines around the objects, making it easier to recognize and distinguish them.

In essence, edge detection helps computers "see" and understand the structure of objects in an image by focusing on their outlines. This is useful for many applications, from identifying objects in a photo to helping robots navigate by recognizing obstacles.

Its Types:

1. Canny Edge Detection using CNNs

   - CNNs can be trained to replicate and enhance the traditional Canny edge detector's functionality. This involves learning to perform Gaussian blurring, gradient calculation, non-maximum suppression, and edge tracking.

2. Holistically-Nested Edge Detection (HED)

   - HED uses deep learning to learn rich hierarchical features for edge detection. It combines deep supervision with side outputs from different convolutional layers, capturing edges at multiple scales and achieving state-of-the-art results.

3. Structured Edges Detection

   - This method involves training a CNN to predict structured edge maps by considering contextual information. It usually involves a combination of deep learning with structured learning approaches to enhance edge detection accuracy.

4. DeepEdge

   - Utilizes CNNs to extract features from multiple scales and then combines these features to detect edges. It focuses on capturing both local and global context for accurate edge detection.

5. Richer Convolutional Features (RCF)

   - RCF enhances edge detection by fusing features from different convolutional layers in a CNN. It leverages the rich hierarchical information present in different layers to improve edge detection performance.

These CNN-based methods are important as they leverage the powerful feature extraction capabilities of deep learning, leading to more accurate and robust edge detection compared to traditional methods.

TREE CROWN DETECTION:

Tree crown detection is a process in computer vision and remote sensing used to identify and delineate the tops of trees, known as tree crowns, in aerial or satellite images. Think of it as finding and marking the outer edges of the leafy part of trees when viewed from above.

Understand of Tree Crown Detection:

1. Imagine an Aerial View: Picture a photograph taken from an airplane or satellite showing a forest.
2. Focusing on Tree Tops: Tree crown detection focuses on identifying the areas in the image where the tops of trees are visible.
3. Outlining Tree Crowns: The process involves drawing outlines around the tree crowns, separating one tree from another, and from the ground or other objects.
4. Using Computer Vision: Specialized algorithms and models, like Convolutional Neural Networks (CNNs), are used to automatically detect these tree crowns based on their shape, color, and texture.

Tree crown detection is important for several reasons:

- Forestry Management: Helps in monitoring forest health, assessing resources, and planning for sustainable land use.
- Ecological Studies: Aids in studying biodiversity by quantifying and analyzing tree populations.
- Climate Research: Supports the estimation of carbon sequestration rates, crucial for climate change mitigation

In essence, tree crown detection uses advanced technology to help scientists and environmentalists accurately map and monitor forests from above.

SHP corrector plugin : Aim to reduce human intervention and focus on making the automated part accurate and precise.

rough aerial surveys, satellite imagery, or photography.

- Preprocess images: resizing, normalization, and augmentation.
- Annotate images to create ground truth data for supervised learning.

2. Model Architecture Design:

- Use U-Net architecture with encoding and decoding paths for image segmentation.
- Fine-tune depth, width, and layer configurations for optimal performance.

3. Loss Function Selection:

- Use Binary Cross-Entropy Loss to measure dissimilarity between predicted and ground truth segmentation masks.

4. Model Training:

- Feed annotated images into the U-Net model.
- Optimize parameters using algorithms like SGD, Adam, or RMSprop.
- Use backpropagation to iteratively enhance the model's accuracy in segmenting tree crowns.

U-NET model:

The U-Net model is a type of Convolutional Neural Network (CNN) designed specifically for image segmentation tasks, where the goal is to classify each pixel in an image into a specific category. It is particularly effective in medical imaging and other fields requiring precise localization and segmentation.

Structure of U-Net

1. U-Shaped Architecture: The model gets its name from its U-shaped structure, which consists of two main parts: the encoder (downsampling path) and the decoder (upsampling path).

2. Encoder (Contracting Path):

- Purpose: To capture the context in the image by progressively reducing the spatial dimensions and increasing the depth.
- Layers: It consists of repeated application of two 3x3 convolution layers (with ReLU activation) followed by a 2x2 max-pooling operation for downsampling.
- Result: This path creates a compressed representation of the input image, capturing the essential features.

3. Bottleneck:

- Purpose: Acts as a bridge between the encoder and decoder.

- Layers: It usually consists of two 3x3 convolutions followed by a 2x2 upsampling operation.

4. Decoder (Expansive Path):

- Purpose: To recover the spatial dimensions while retaining the context information, ultimately producing a segmented image of the same size as the input.

- Layers: It involves upsampling (using transposed convolutions or up-convolutions) followed by concatenation with corresponding feature maps from the encoder (skip connections), and two 3x3 convolutions with ReLU activation.

- Result: This path helps in refining the features and bringing the output back to the original image size.

5. Skip Connections:

- Purpose: To combine low-level features from the encoder with high-level features from the decoder.

- Layers: These are direct connections between corresponding layers of the encoder and decoder.

- Result: They help in preserving spatial information that may be lost during downsampling.

6. Output Layer:

- Purpose: To produce the final segmented output.

- Layers: A final 1x1 convolution layer that reduces the number of feature maps to the number of classes, followed by a softmax or sigmoid activation function to produce class probabilities for each pixel.

<u>Why U-Net is Effective:</u>

1. Localization and Context: The U-shaped architecture allows the network to localize features precisely by using the context captured in the encoder and the fine details captured in the decoder.

2. Efficient Training: The skip connections enable better gradient flow during backpropagation, making the network easier to train.

3. High Accuracy: By combining coarse and fine features through skip connections, U-Net achieves high accuracy in pixel-level classification tasks.

4. Versatility: U-Net is versatile and can be used for various segmentation tasks beyond medical imaging, such as satellite image analysis, object detection in images, and more.

In essence, the U-Net model is highly effective for tasks where precise pixel-level classification is needed. Its unique architecture, combining down sampling and up sampling paths with skip connections, allows it to excel in producing detailed and accurate segmentation maps.

5. Hyperparameter Tuning:

- Optimization of parameters like learning rate, batch size, optimizer choice, and regularization techniques.
- Experimentation and validation on the validation set to find the optimal balance between underfitting and overfitting.

6. Model Evaluation:

- Evaluation using the test set.
- Metrics include Intersection over Union (IoU), F1 score, precision, recall, and accuracy.
- Higher IoU, F1 score, precision, and recall indicate better segmentation performance.

7. Iterative Improvement:

- Ongoing process involving analysis of misclassified instances.
- Addressing dataset biases, fine-tuning parameters, or augmenting the dataset with additional annotated images for improvement.

8. Deployment and Operationalization:

- Deployment into the operational environment.
- Integration into software applications, GIS platforms, or deployment on cloud servers or edge devices.
- Capable of performing real-time or batch tree crown detection efficiently.

9. Maintenance and Updates:

   - Continuous monitoring and maintenance.

   - Regular retraining with new data, updating model architecture or parameters, and addressing performance drift.

In conclusion, training a U-Net model for tree crown detection involves a structured approach encompassing data collection, preprocessing, architecture design, hyperparameter tuning, evaluation, deployment, and continuous improvement. This process results in a robust and accurate model crucial for forestry management and environmental monitoring.

SYSTEM DESIGN

High-Level Design

The high-level design of the system involves creating a big-picture view of the architecture, including hardware, database structure, application layers, navigation flow, security, and technology frameworks. This is akin to drawing a blueprint for the entire system, ensuring all components fit together seamlessly and serve user and business needs effectively.

U-Net Architecture for Image Size 256x256

For processing images sized 256x256, the U-Net model is structured as follows:

Encoder:

- Input Layer (256x256x3): Takes a 256x256 image with three color channels (RGB).
- Down sampling Path: Uses convolutional layers to extract high-level features, with pooling layers reducing spatial dimensions to create a compressed representation of the image.

Intermediate Bottleneck:

- Bottleneck: The deepest part of the network, capturing the most abstract features of the input image.

Decoder:

- Upsampling Path: Utilizes transposed convolutional layers to increase the spatial dimensions of feature maps, gradually reconstructing the original image size.

- Skip Connections: Direct links between corresponding layers in the encoder and decoder, preserving high-resolution details for precise segmentation.

- Concatenation: Combines feature maps from the encoder with those being upsampled in the decoder, ensuring both low-level and high-level features are used for accurate segmentation.

Output Layer:

- Output Layer (256x256xN): Produces a final segmentation map with N channels, where N represents the number of segments or classes. Each pixel in this output corresponds to a specific segment, such as a tree crown.

The U-Net's design, with its combination of encoder-decoder pathways and skip connections, allows it to capture detailed and broad features effectively, making it ideal for tasks like tree crown detection in 256x256 images.

Introduction to Neural Networks

Neural networks are a core concept in machine learning, inspired by how the human brain works. They consist of interconnected nodes, or neurons, organized in layers. These networks can recognize patterns, learn from data, and make decisions without needing explicit instructions for each task. Neural networks have revolutionized many industries, from image recognition and language processing to autonomous vehicles and healthcare.

Key Points of Neural Networks:

1. Basic Structure:

   - Neurons (Nodes): The basic units that receive input, process it, and output results.

   - Layers: Include an input layer (receives data), hidden layers (process data), and an output layer (produces the final result).

   - Connections (Edges): Neurons are connected by weights, influencing how information flows.

2. Feedforward Neural Networks:

   - Activation Function: Each neuron uses an activation function like ReLU to introduce non-linearity, helping the network learn complex patterns.

- Feedforward Process: Data flows from input to output, with each layer processing and passing on the information.

- Output: The final result is compared to the desired output during training to improve accuracy.

3. Training Neural Networks:

- Loss Function: Measures the difference between predicted and actual results, guiding the training process to minimize errors.

- Backpropagation: Adjusts weights based on the loss function, using algorithms like gradient descent.

- Epochs and Batch Training: Training is done in cycles (epochs) and smaller data batches to improve efficiency.

4. Types of Neural Networks:

- Multilayer Perceptrons (MLP): Basic feedforward networks.

- Convolutional Neural Networks (CNN): Specialized for image processing.

- Recurrent Neural Networks (RNN): Designed for sequential data like time series.

- LSTM and GRU: Variants of RNNs that handle long-term dependencies better.

- Generative Adversarial Networks (GAN): Consist of two networks (generator and discriminator) used for generating new data.

Convolutional Neural Network (CNN):

CNNs are a type of neural network especially good at processing visual data like images. They can recognize patterns and details in images, making them essential for tasks like image recognition, classification, and object detection. CNNs mimic how the human visual system works, learning features from images automatically.

Key Components of CNN Architecture:

1. Convolutional Layer:

- Uses filters (small matrices) to scan over the input image and detect patterns like edges.

2. Activation Function:

- Applies functions like ReLU to introduce non-linearity, helping the network learn more complex patterns.

3. Pooling Layer:

- Reduces the size of feature maps by selecting the most important information, making the network more efficient.

4. Fully Connected Layer:

- After convolutional and pooling layers, neurons are fully connected to each other, helping with high-level reasoning and predictions.

5. Output Layer:

- Produces the final result, often using softmax activation to convert outputs into probabilities for classification tasks.

6. Flattening:

- Transforms the high-level features from the previous layers into a vector that can be processed by the fully connected layers.

7. Training CNNs:

- Uses labeled data and backpropagation to adjust weights and minimize errors, typically employing optimization algorithms like stochastic gradient descent (SGD).

In summary, CNNs are powerful tools for visual data processing, capable of learning detailed and complex features from images. Their ability to handle and interpret visual information has led to significant advancements in fields like healthcare, autonomous vehicles, and many other applications involving image analysis.

Summary of Tree Crown Detection Model Development

This project aims to develop a machine learning model for detecting tree crowns in satellite images of forests. The dataset used in this project has been meticulously curated to support research in computer vision and remote sensing. It includes a diverse set of RGB satellite images annotated with precise masks around individual tree crowns, covering various geographical locations, seasons, and weather conditions. This diversity ensures a robust dataset that enhances the model's ability to generalize well across different settings. Ethical considerations were strictly followed in the dataset's creation, ensuring compliance with privacy and legal regulations.

Dataset Preparation

The dataset preparation process involves acquiring and organizing the satellite images and their corresponding masks. These images are divided into three sets: training, validation, and testing. The training set is utilized to train the model, the validation set is used for hyperparameter tuning and model selection, and the testing set is employed to assess the model's performance on unseen data. This structured division ensures that the model can generalize effectively to new data, a critical aspect for any machine learning model.

Drive Mounting and Library Imports

The first step in the coding process is mounting Google Drive to access the dataset. This step is essential for loading and saving data, as the dataset and model outputs are stored on Google Drive. Following this, various libraries and functions are imported. These include essential tools for data preprocessing, model building, and evaluation. Libraries such as TensorFlow, Keras, and Albumentations are used for deep learning and data augmentation, while NumPy and Pandas are utilized for data manipulation.

Configuration Setup

Configuration variables are defined to streamline the data preprocessing and model training processes. These variables include paths to the dataset, image size, batch size, and other parameters. Setting these configurations helps maintain consistency and ease of modification throughout the development process.

Data Preparation and Helper Functions

A helper function is defined to prepare the images and masks for model training. This function reads the files, decodes the images, and converts them into tensor format suitable for TensorFlow. The images are resized to a standard dimension and normalized to ensure consistent input for the model. Partial functions are then created to streamline the data preparation process for both images and masks. These functions are used to load and preprocess the data in a consistent manner.

Data Loaders

Data loaders are created for the images and masks, which are then combined into a single dataset. This dataset is split into training and testing sets based on a predefined ratio. The training data is further prepared by shuffling, repeating, and batching to ensure that the model receives a diverse and well-distributed set of training examples.

Model Architecture: U-Net

The U-Net architecture is chosen for this project due to its effectiveness in segmentation tasks. U-Net consists of an encoder and a decoder. The encoder comprises several convolutional blocks and max-pooling layers, which extract features from the input images. The decoder consists of upsampling layers and concatenations with the corresponding encoder layers, helping to reconstruct the segmented image from the extracted features.

Convolutional Block

A convolutional block is defined to serve as the building block for both the encoder and the decoder. It consists of two convolutional layers followed by batch normalization and ReLU activation. This block helps in feature extraction while maintaining the spatial dimensions of the input.

Encoder

The encoder part of the U-Net model is built using convolutional blocks and max-pooling layers. Each block extracts features from the input images and reduces the spatial dimensions through pooling, which helps in capturing higher-level features.

Decoder

The decoder part reconstructs the segmented image from the encoded features. It uses upsampling layers to increase the spatial dimensions and concatenates the upsampled features with the corresponding features from the encoder. This skip connection helps in retaining spatial information lost during downsampling.

Model Assembly

The U-Net model is assembled by connecting the encoder and decoder parts. The final output layer uses a sigmoid activation function to produce a binary mask, indicating the presence or absence of tree crowns.

Model Evaluation

Custom evaluation metrics, such as Intersection over Union (IoU) and F1 score, are defined to measure the model's performance. These metrics provide a comprehensive understanding of the model's accuracy and its ability to correctly identify tree crowns.

Training Process

The U-Net model is compiled and trained using the defined data loaders and callbacks. The training process involves optimizing the model weights to minimize the loss function, which in this case is binary cross-entropy. Callbacks like ModelCheckpoint and EarlyStopping are used to save the best model and prevent overfitting.

Results Visualization

After training, the model's performance is visualized by plotting the training and validation loss curves. Additionally, predictions on test samples are visualized to assess the model's ability to segment tree crowns accurately.

Conclusion

This project demonstrates a comprehensive approach to developing a machine learning model for tree crown detection using satellite images. The detailed dataset preparation, robust U-Net architecture, and careful evaluation metrics contribute to creating a model that can significantly aid in forestry management, biodiversity assessment, and environmental monitoring. By leveraging this dataset and model, researchers can further the efforts in sustainable forestry practices and environmental conservation.

Briefed by,

Sanskriti Bhardwaj & Vanshika Pandey