DEEP LEARNING

May 2024

MINI Project Report

Semantic Segmentation for Self-Driving-Cars

- SEHILI Chaima
- HACHOUD Mohammed
- Frihaoui Ayoub
- Belagha Ayoub

SEMANTIC SEGMENTATION FOR SELF-DRIVING-CARS

Image segmentation plays a vital role in **autonomous driving.** Self-driving cars need appropriate pixel knowledge about the surroundings to drive without accidents. Therefore, image segmentation can assist in recognizing lanes, traffic lights, highways, street signs, cross marks, other cars, pedestrians, and other essential information.

Semantic segmentation:

this involves arranging the pixels in an image based on semantic class.

Instance Segmentation:

this technique involves classifying pixels based on the instances of an object instead of classes.

Panoptic Segmentation:

a combination of semantic and instance segmentation. It predicts the identity of each object, separating every instance of each object in the image.

SEMANTIC SEGMENTATION FOR SELF-DRIVING-CARS:

Semantic segmentation is crucial for self-driving cars as it enables them to understand and interpret the environment from sensor data, especially from cameras. It divides the scene into segments, identifying objects like lanes, pedestrians, and vehicles, essential for tasks like localization, mapping, path planning, navigation, object detection, and decision-making. Ultimately, semantic segmentation is fundamental for autonomous vehicles to operate safely and effectively in various driving conditions.

ABOUT DATASET

CONTEXT

Cityscapes data contains labeled videos taken from vehicles driven in Germany. This version is a processed subsample created as part of the Pix2Pix paper. The dataset has still images from the original videos, and the semantic segmentation labels are shown in images alongside the original image. This is one of the best datasets around for semantic segmentation tasks.

CONTENT

This dataset has 2975 training images files and 500 validation image files. Each image file is 256x512 pixels, and each file is a composite with the original photo on the left half of the image, alongside the labeled image (output of semantic segmentation) on the right half.

DATA PREPARATION & GENERATION

1-LOADING IMAGES AND MASKS

Using the **LoadImage** function: It loads the image using Pillow (PIL) library and converts it into a NumPy array.
The image is **sliced to obtain the RGB** channels (first 256 columns), while the mask is sliced to obtain segmentation information (from column 256 onwards).
The function **returns the image and its corresponding mask.**

2-K-MEANS CLUSTERING FOR SEGMENTATION

To find the most ipmortant colors and identify similar colors we can use **KMeans** clustering. This also allows us to go from a color representation to a class representation.

3-COLOR ASSIGNMENT FOR VISUALIZATION:

Assigns colors to the segmented image based on the predicted segmentation labels.

4-DATA GENERATION:

It yields NumPy arrays of images and segmentation masks in batches.

PART ONE
SEGNET

SEGNET

SegNet is a deep learning model designed for semantic segmentation in computer vision, dividing images into segments and assigning class labels to pixels. Its encoder-decoder structure captures image features and reconstructs segmented output, with unique use of pooling indices for precise upsampling. Widely applied in fields like autonomous driving and medical imaging, SegNet helps vehicles interpret their surroundings by identifying and segmenting road elements such as lanes, pedestrians, vehicles, signs, and obstacles, aiding in safe navigation.

The SegNet architecture consists of an **encoder-decoder network** designed specifically for semantic segmentation tasks in computer vision. Here's a breakdown of its components:

1. **Encoder**: it captures hierarchical features of input images through convolutional layers, extracting abstract representations and encoding spatial information and local features.
2. **Decoder:** it reconstructs segmented output from encoded features. It utilizes upsampling layers to gradually increase spatial resolution, aiding in the creation of a high-resolution segmentation mask.
3. **Pooling Indices:** SegNet employs pooling indices acquired during max-pooling in the encoder phase. These indices, signifying maximum activation locations within pooling regions, are stored and reused in the decoder phase for precise upsampling, preserving spatial information lost during downsampling.
4. **Softmax Layer:** At the end of the decoder, SegNet typically uses a softmax layer to assign class probabilities to each pixel in the segmented output. This layer produces a probability distribution over the classes for each pixel, enabling pixel-wise classification.

TRAINING:

Here's a summary of what we did on the training of segnet

1 - Import Callbacks: we add two callback functions from **TensorFlow .keras.callbacks** module: **EarlyStopping** to prevents overfitting and saves computational resources. and **ReduceLROnPlateau for** leading the Model to better convergence

2- Loss Function: Categorical crossentropy is a popular choice for multi-class classification tasks, including semantic segmentation.

3- Optimizer: Adam is a widely used optimizer in deep learning. It adapts the learning rate for each parameter during training, which can lead to faster convergence and better performance compared to traditional optimization methods like stochastic gradient descent (SGD).

4- Number of Epochs: The training is configured to run for 30 epochs.

EVALUATION:

It's impressive that the model achieved a **test accuracy of 84%**, indicating its ability to effectively classify pixels into different classes. Notably, it successfully segmented all parts of the images, including trees, roads, and cars. This comprehensive segmentation demonstrates the model's capability to understand and delineate various elements within the scene, a crucial aspect for tasks like autonomous driving or scene understanding in computer vision applications. Achieving accurate segmentation across diverse objects and backgrounds is a significant milestone.

PART TWO: UNET

UNET

Initially designed for biomedical image segmentation with limited datasets, **UNet** is now employed in self-driving cars for accurate semantic segmentation tasks, especially from camera sensor data. Its proficiency in precisely segmenting and localizing objects contributes to the perception pipeline, empowering vehicles to understand and navigate varied real-world environments securely and autonomously.

The architecture of UNet is characterized by a symmetric encoder-decoder structure with skip connections. Here's a brief overview of its components:

1. **<u>Encoder:</u>** The encoder in UNet comprises convolutional layers followed by max-pooling layers, gradually reducing the spatial dimensions of the input image and capturing hierarchical features.
2. **<u>Decoder:</u>** The decoder in UNet mirrors the encoder's structure, consisting of upsampling layers such as transposed convolutions or upsampling followed by convolutional layers. It gradually restores the spatial dimensions to the original input size while reconstructing the segmented output.
3. **<u>Skip Connections:</u>** UNet features skip connections linking corresponding encoder and decoder layers, retaining fine-grained spatial information from the encoder's higher-resolution feature maps. This aids in precise object localization during segmentation.
4. **<u>Final Layer:</u>** The final layer of UNet usually consists of a convolutional layer with a softmax activation function. It generates a pixel-wise probability map for each class in the segmentation task, facilitating detailed segmentation of objects in the image.

TRAINING:

Here's a summary of what we did on the training of segnet

1 - Import Callbacks: we add two callback functions from **TensorFlow .keras.callbacks** module: **EarlyStopping** to prevents overfitting and saves computational resources. and **ReduceLROnPlateau for** leading the Model to better convergence

2- Loss Function: Categorical crossentropy is a popular choice for multi-class classification tasks, including semantic segmentation.

3- Optimizer: Adam is a widely used optimizer in deep learning. It adapts the learning rate for each parameter during training, which can lead to faster convergence and better performance compared to traditional optimization methods like stochastic gradient descent (SGD).

4- Number of Epochs: The training is configured to run for 30 epochs.

EVALUATION:

It's impressive that the model achieved a **test accuracy of 85%**, indicating its ability to effectively classify pixels into different classes. Notably, it successfully segmented all parts of the images, including trees, roads, and cars. This comprehensive segmentation demonstrates the model's capability to understand and delineate various elements within the scene, a crucial aspect for tasks like autonomous driving or scene understanding in computer vision applications. Achieving accurate segmentation across diverse objects and backgrounds is a significant milestone.

UNET & MOBILENETV2

UNet with MobileNetv2 combines the UNet architecture with MobileNetv2 as the backbone network for feature extraction. It's used for semantic segmentation, aiming to assign class labels to individual pixels in images. Its significance in self-driving cars lies in its ability to accurately segment objects like roads, vehicles, and pedestrians from camera images, enhancing the perception pipeline of autonomous vehicles. **MobileNetv**2's efficiency allows for real-time processing, crucial for quick reactions to dynamic driving conditions. Overall, UNet with **MobileNetv2** enhances the visual perception of self-driving cars, contributing to their safety and autonomy.

UNet with MobileNetv2 architecture typically comprises the following components:

1. **<u>Backbone Network (MobileNetv2):</u>** MobileNetv2 serves as the backbone network for feature extraction. It efficiently extracts hierarchical features from input images while minimizing computational resources, making it suitable for real-time applications.
2. **<u>Encoder-Decoder Structure (UNet):</u>** UNet's encoder-decoder architecture is used for semantic segmentation tasks. The encoder extracts features from the input image, while the decoder reconstructs the segmented output. Skip connections may be employed to retain spatial information and improve segmentation accuracy.
3. **<u>Upsampling Layers:</u>** The decoder typically includes upsampling layers, such as transposed convolutions or bilinear upsampling, followed by convolutional layers. These layers gradually increase the spatial resolution of feature maps to produce the final segmentation output.
4. **<u>Final Layer:</u>** The final layer of the network typically consists of a convolutional layer with a softmax activation function. This layer produces a pixel-wise probability map for each class in the segmentation task.

TRAINING:

Here's a summary of what we did on the training of segnet

1 - Import Callbacks: we add two callback functions from **TensorFlow .keras.callbacks** module: **EarlyStopping** to prevents overfitting and saves computational resources. and **ReduceLROnPlateau for** leading the Model to better convergence

2- Loss Function: Categorical crossentropy is a popular choice for multi-class classification tasks, including semantic segmentation.

3- Optimizer: Adam is a widely used optimizer in deep learning. It adapts the learning rate for each parameter during training, which can lead to faster convergence and better performance compared to traditional optimization methods like stochastic gradient descent (SGD).

4- Number of Epochs: The training is configured to run for 30 epochs.

EVALUATION:

Achieving a **test accuracy of 82% on Unet with MobileNetv2 after just 30 epochs** is impressive. It indicates that the model has learned to effectively segment objects in the input images, showcasing its capability to understand complex visual scenes. With further training, the model may continue to improve and potentially achieve even higher accuracy, highlighting the effectiveness Unet architecture combined with the efficient feature extraction capabilities of MobileNetv2 for semantic segmentation tasks.

OUR U-NET ARCHITECTURE:

1. **Encoder**:

- It comprises **five convolutional blocks** . Each block consists of two convolutional layers followed by optional dropout and max-pooling layers.
- In this implementation, the number of filters doubles with each block.

2. **Decoder:**

- It consists of **four upsampling** blocks . Each block performs upsampling using transposed convolution (Conv2DTranspose) followed by concatenation with the corresponding feature map from the contracting path, followed by two convolutional layers.
- The number of filters is halved with each block.

3. **Final Layers:**

- After the decoder, there's a single convolutional layer (Conv2D) followed by a convolutional layer with a kernel size of 1, which acts as a pixel-wise classifier for segmentation.

The provided architecture diverges from the basic U-Net model through the incorporation of dropout layers in select convolutional blocks (cblock3, cblock4, cblock5), allowing for regularization during training. Additionally, it introduces a dynamic approach to filter scaling, doubling the number of filters with each contracting block (cblock1 to cblock5) and halving them with each expanding block (ublock6 to ublock9), enhancing the network's capacity for capturing complex features across different scales. Furthermore, it offers the flexibility to toggle max-pooling in the final convolutional block (cblock5), accommodating scenarios where preserving fine-grained spatial information is paramount. Lastly, by enabling the specification of transposed convolution kernel sizes in the upsampling blocks (upsampling_block), the architecture provides greater customization options, empowering experimentation with various kernel sizes for improved feature reconstruction.

LEARNING CURVES:

EVALUATION:

IIn our customized U-net model we have achieved an impressive test accuracy of **approximately 84%,** showcasing the effectiveness of our customized methodology. This level of accuracy closely aligns with the performance of established architectures like U-Net,

PART THREE
FCN

FCN

FCN stands for Fully Convolutional Network, which is a type of deep learning architecture designed for semantic segmentation tasks. Unlike traditional convolutional neural networks (CNNs) that are typically used for image classification, FCNs can produce dense pixel-wise predictions by preserving spatial information throughout the network. FCNs are powerful tools for semantic segmentation tasks in self-driving cars, enabling vehicles to understand and navigate complex real-world environments by providing dense pixel-wise predictions of the scene.

The architecture of an FCN (Fully Convolutional Network) typically consists of three main components:

1. **<u>Encoder:</u>** The encoder component comprises several convolutional layers responsible for extracting features from the input image. These convolutional layers typically use filters of different sizes to capture hierarchical features at various levels of abstraction. The encoder gradually reduces the spatial dimensions of the input image while preserving important features.
2. **<u>Decoder:</u>** The decoder component reconstructs the segmentation map from the feature maps generated by the encoder. It usually consists of upsampling layers, such as transposed convolutions or upsampling followed by convolutional layers, to gradually increase the spatial dimensions of the feature maps. This process helps in restoring the spatial resolution lost during the downsampling operation in the encoder. Skip connections, which connect corresponding encoder and decoder layers, are often employed to retain fine-grained spatial information and aid in precise localization.
3. **<u>Final Layer:</u>** The final layer of the FCN typically consists of a convolutional layer with a softmax activation function. This layer produces a pixel-wise probability map for each class in the segmentation task. Each pixel in the output map represents the probability of belonging to a specific class, allowing for detailed semantic segmentation of the input image.

TRAINING:

Here's a summary of what we did on the training of segnet

1 - Import Callbacks: we add two callback functions from **TensorFlow .keras.callbacks** module: **EarlyStopping** to prevents overfitting and saves computational resources. and **ReduceLROnPlateau for** leading the Model to better convergence

2- Loss Function: Categorical crossentropy is a popular choice for multi-class classification tasks, including semantic segmentation.

3- Optimizer: Adam is a widely used optimizer in deep learning. It adapts the learning rate for each parameter during training, which can lead to faster convergence and better performance compared to traditional optimization methods like stochastic gradient descent (SGD).

4- Number of Epochs: The training is configured to run for 30 epochs.

EVALUATION:

The FCN model achieved a **test accuracy of 71%**, but increasing the number of epochs beyond 30 can lead to further convergence and higher accuracy. This means that by allowing the model to train for more epochs, it can continue to learn from the data and improve its performance on the task. This is a common phenomenon in deep learning, where extended training time often leads to better model generalization and higher accuracy on unseen data. Therefore, extending the training duration beyond **30 epochs** may result in the FCN model achieving a **higher test accuracy than 71%.**

PART FOUR: DEEPLABV3

WITH RESNET50:

DeepLab is a cutting-edge deep learning architecture crafted for semantic image segmentation tasks, originating from Google researchers in 2016. It excels in dense pixel-wise classification while maintaining spatial details via atrous convolutions and **conditional random fields (CRFs)**. Particularly in self-driving cars, DeepLab proves potent, accurately segmenting objects at the pixel level. Its preservation of spatial information enhances understanding of intricate driving environments, crucial for safe autonomous navigation.

DeepLabv3 with ResNet50 as the backbone network combines the DeepLabv3 architecture with the ResNet50 backbone for semantic segmentation tasks. Here's an overview of the architecture:

1. **Backbone Network (ResNet50):** ResNet50 serves as the backbone network in DeepLabv3, facilitating feature extraction. It's renowned for its depth and skip connections, mitigating the vanishing gradient issue and enabling robust feature learning. ResNet50 extracts hierarchical features from input images.
2. **Atrous Spatial Pyramid Pooling (ASPP):** DeepLabv3 utilizes Atrous Spatial Pyramid Pooling (ASPP) to capture multi-scale information efficiently. ASPP integrates parallel atrous convolutional layers with varying dilation rates, enabling simultaneous analysis of objects at multiple scales. This aids in accurately segmenting objects of diverse sizes in the input image.
3. **Decoder: DeepLabv3 with ResNet50** DeepLabv3 typically incorporates a decoder module to refine segmentation results and restore spatial resolution. This decoder often comprises upsampling layers like transposed convolutions or bilinear upsampling, followed by convolutional layers. Skip connections may be employed to integrate low-level features from the encoder, enhancing segmentation accuracy.
4. **Conditional Random Fields (CRFs):** Some variants of DeepLabv3 may apply CRFs as a post-processing step to refine segmentation boundaries and improve the spatial coherence of the segmentation results. CRFs help in reducing segmentation artifacts and producing smoother segmentation maps.

TRAINING:

Here's a summary of what we did on the training of segnet

1 - Import Callbacks: we add two callback functions from **TensorFlow .keras.callbacks** module: **EarlyStopping** to prevents overfitting and saves computational resources. and **ReduceLROnPlateau for** leading the Model to better convergence

2- Loss Function: Categorical crossentropy is a popular choice for multi-class classification tasks, including semantic segmentation.

3- Optimizer: Adam is a widely used optimizer in deep learning. It adapts the learning rate for each parameter during training, which can lead to faster convergence and better performance compared to traditional optimization methods like stochastic gradient descent (SGD).

4- Number of Epochs: The training is configured to run for 30 epochs.

EVALUATION:

Achieving a **test accuracy of 83% on DeepLabv3 with ResNet50 after just 30 epochs** is a commendable result. It indicates that the model has learned to effectively segment objects in the input images, demonstrating its capability to understand complex visual scenes. With further training, the model may continue to improve and potentially achieve even higher accuracy, highlighting the effectiveness of DeepLabv3 architecture combined with the ResNet50 backbone for semantic segmentation tasks.

WITH MOBILENETV2:

DeepLabv3 with MobileNetv2 is designed for semantic segmentation tasks, aiming to assign class labels to individual pixels in images. Its application in self-driving cars is significant as it accurately segments objects in visual scenes captured by onboard cameras. This enhances the perception pipeline of autonomous vehicles, enabling them to understand the environment, identify key elements, and make informed navigation decisions. The efficiency of MobileNetv2 allows for real-time processing, crucial for prompt reactions to dynamic driving conditions. Overall, DeepLabv3 with MobileNetv2 enhances the visual perception of self-driving cars, contributing to their safety and autonomy.

DeepLabv3 with MobileNetv2 architecture comprises the following components:

1. **Backbone Network (MobileNetv2)**: MobileNetv2 serves as the backbone network for feature extraction. It is a lightweight convolutional neural network architecture optimized for mobile and embedded devices. MobileNetv2 efficiently extracts hierarchical features from input images while minimizing computational resources.
2. **Atrous Spatial Pyramid Pooling (ASPP):** DeepLabv3 incorporates ASPP to capture multi-scale information effectively. ASPP consists of parallel atrous convolutional layers with different dilation rates, allowing the network to analyze objects at multiple scales simultaneously. This helps accurately segment objects of various sizes in the input image.
3. **Decoder:** The decoder module refines segmentation results and restores spatial resolution. It typically includes upsampling layers like transposed convolutions or bilinear upsampling, followed by convolutional layers. Skip connections may be used to integrate low-level features from the encoder, improving segmentation accuracy.
4. **Conditional Random Fields (CRFs):** In some variants, CRFs are applied as a post-processing step to refine segmentation boundaries and improve the spatial coherence of the segmentation results. CRFs help reduce segmentation artifacts and produce smoother segmentation maps.

TRAINING:

Here's a summary of what we did on the training of segnet

1 - Import Callbacks: we add two callback functions from **TensorFlow .keras.callbacks** module: **EarlyStopping** to prevents overfitting and saves computational resources. and **ReduceLROnPlateau for** leading the Model to better convergence

2- Loss Function: Categorical crossentropy is a popular choice for multi-class classification tasks, including semantic segmentation.

3- Optimizer: Adam is a widely used optimizer in deep learning. It adapts the learning rate for each parameter during training, which can lead to faster convergence and better performance compared to traditional optimization methods like stochastic gradient descent (SGD).

4- Number of Epochs: The training is configured to run for 30 epochs.

EVALUATION:

Achieving a **test accuracy of 81% on DeepLabv3 with MobileNetv2 after just 30 epochs** is impressive. It indicates that the model has learned to effectively segment objects in the input images, showcasing its capability to understand complex visual scenes. With further training, the model may continue to improve and potentially achieve even higher accuracy, highlighting the effectiveness of DeepLabv3 architecture combined with the efficient feature extraction capabilities of MobileNetv2 for semantic segmentation tasks.