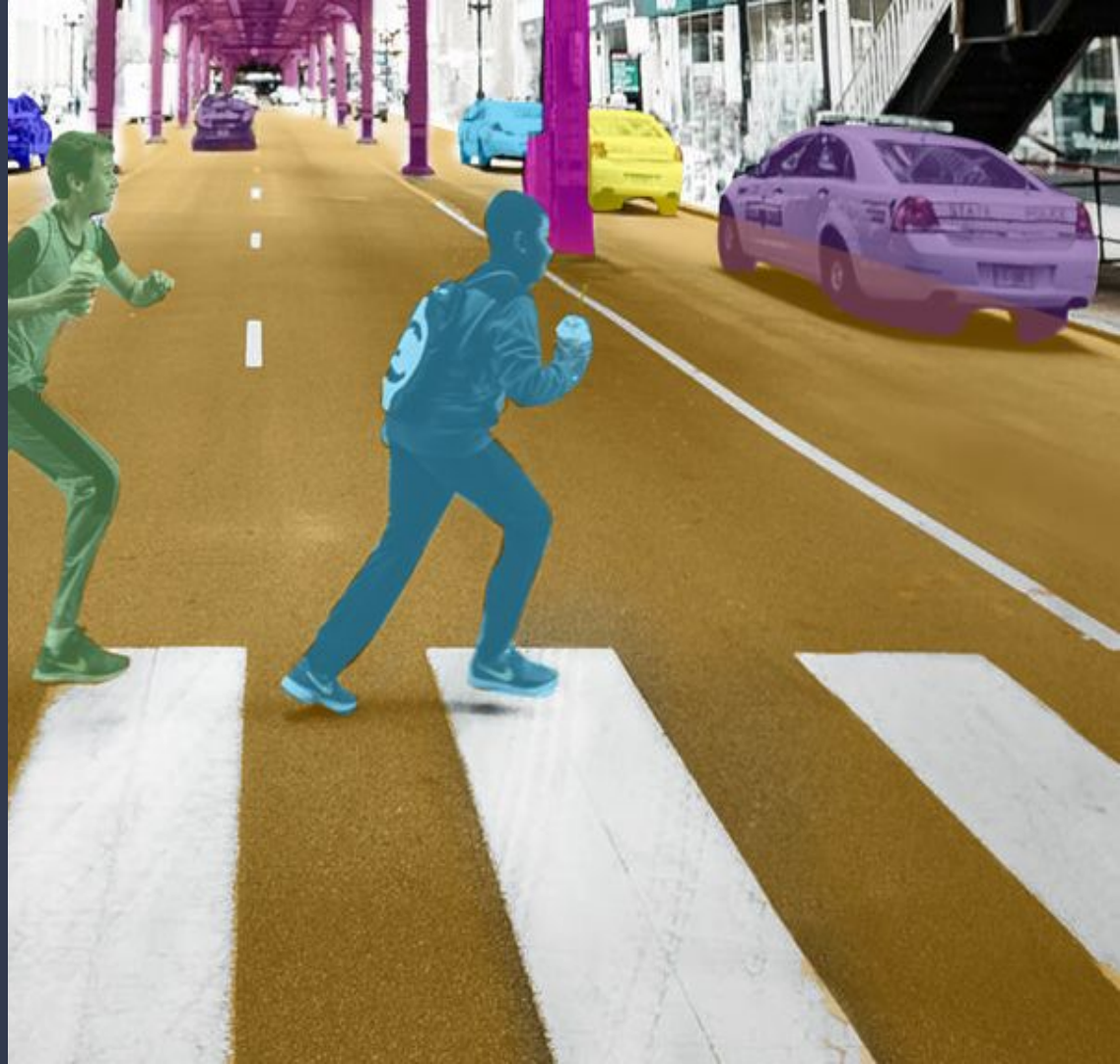
SEMANTIC SEGMENTATION

A model comparison on **Cityscapes Dataset**

AGENDA

1) Introduction
2) Preliminary concepts
3) Models
    a) DeepLabV3
    b) U-Net
    c) DenseASPP
4) Results



Theme

Classification at pixel level

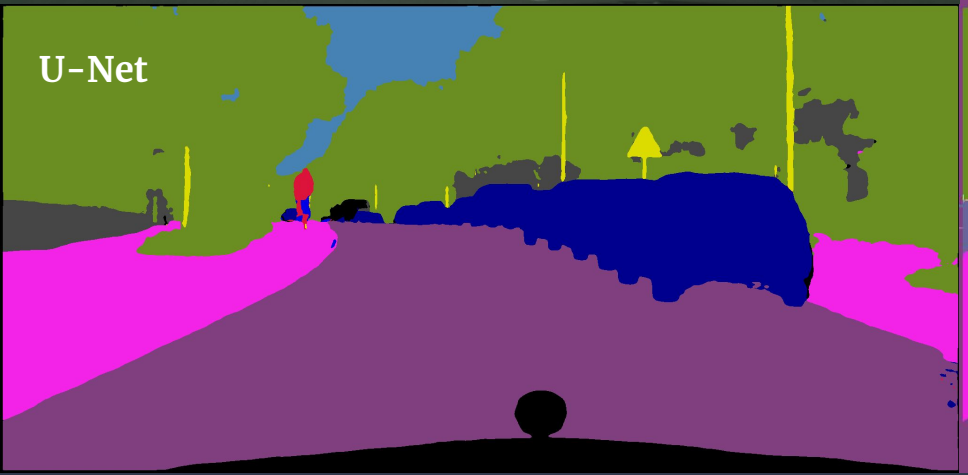
- Assigns labels to each pixel in an image

Creating mask for various objects

- Learns from annotated images

Context and objects emerging

- It is not object detection nor instance segmentation
- Every instance of different objects belong to the same class



Given the same input image different models can obtain different masks with different accuracies

Street scene human and self-driving car

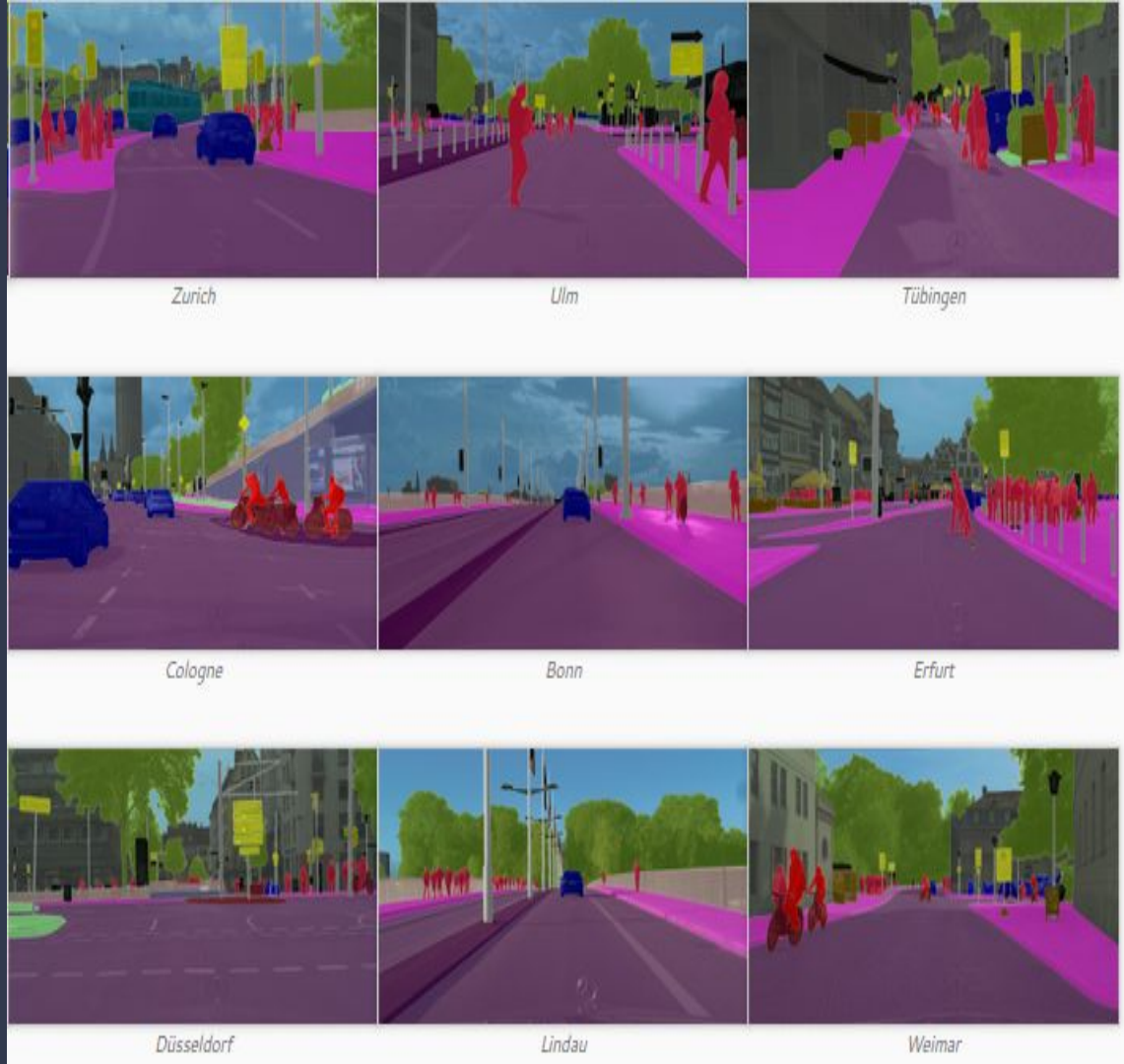
The car records the scene and for each frame processes its mask, assigning one of the possible labels to each pixel

This mask ables the car to make decisions based on the class of the pixels and their location in the frame

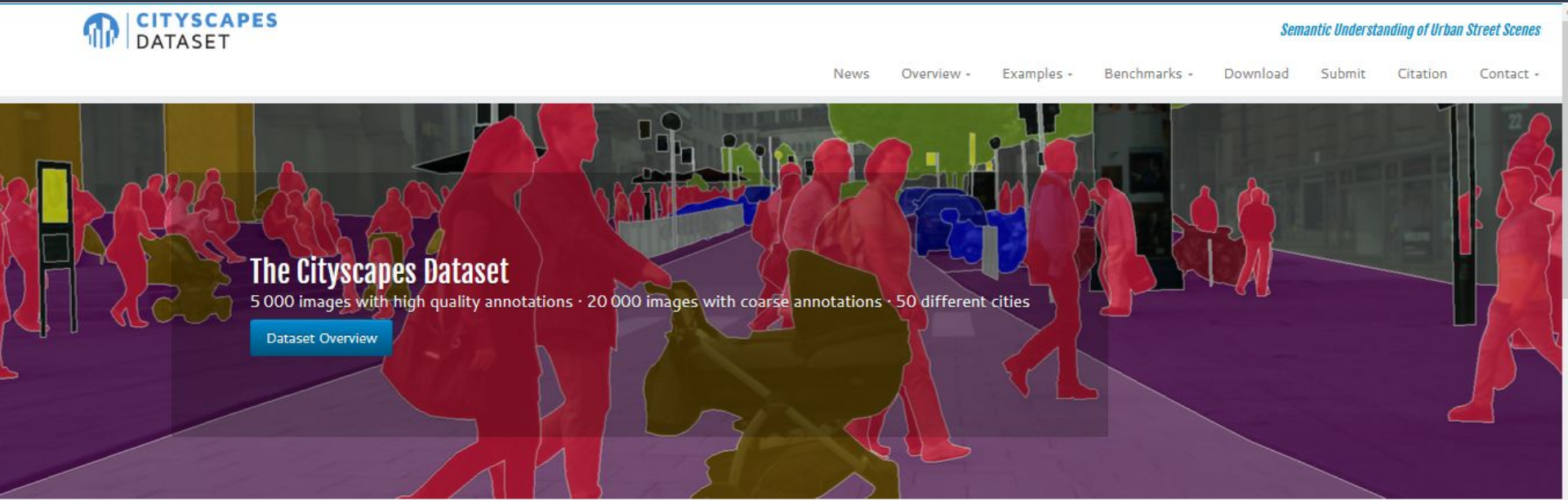
- AUTONOMOUS DRIVING SYSTEMS
- TRAFFIC MONITORING
- NAVIGATION SYSTEMS
- URBAN PLANNING

Dataset

- Cityscapes



Cityscapes



Popular freely available dataset for assessing the performance of vision algorithms for major tasks of semantic urban scene understanding

Features	Number
Annotated Images	5000
Classes	30
German Cities	50
Weather Conditions	Various
Daytime	Various
Months	Various
Metadata	Various

Features

Polygonal annotations

- Dense semantic segmentation
- Instance segmentation for vehicle and people

Complexity

- 30 classes
- See Class Definitions for a list of all classes and have a look at the applied labeling policy.

Diversity

- 50 cities
- Several months (spring, summer, fall)
- Daytime
- Good/medium weather conditions
- Manually selected frames
  - Large number of dynamic objects
  - Varying scene layout
  - Varying background

Volume

- 5 000 annotated images with fine annotations (examples)
- 20 000 annotated images with coarse annotations (examples)

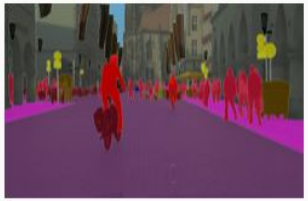
Metadata

- Preceding and trailing video frames. Each annotated image is the 20th image from a 30 frame video snippets (1.8s)
- Corresponding right stereo views
- GPS coordinates
- Ego-motion data from vehicle odometry
- Outside temperature from vehicle sensor

Extensions by other researchers

- Bounding box annotations of people
- Images augmented with fog and rain

Type of annotations



Contained cities

Preliminary concepts

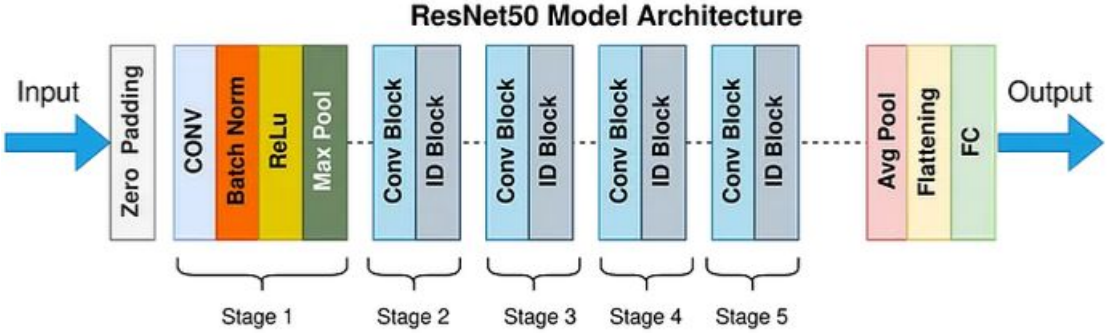
- ResNet50
- Atrous Convolution
- Spatial Pyramid Pooling

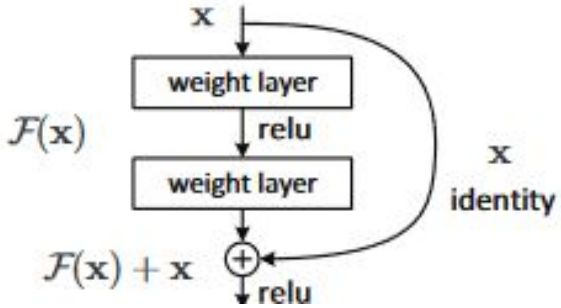
ResNet50

Convolutional neural network used as a **backbone** network in many computer vision tasks, including semantic segmentation.

Contains 50 layers: 1 MaxPool layer, 1 Average Pool layer and 48 convolutional layers

It begins with a convolutional layer with 64 filters 7x7 and a stride of 2, followed by a rectified linear unit (ReLU) and a 3x3 max pooling layer and a stride of 2.



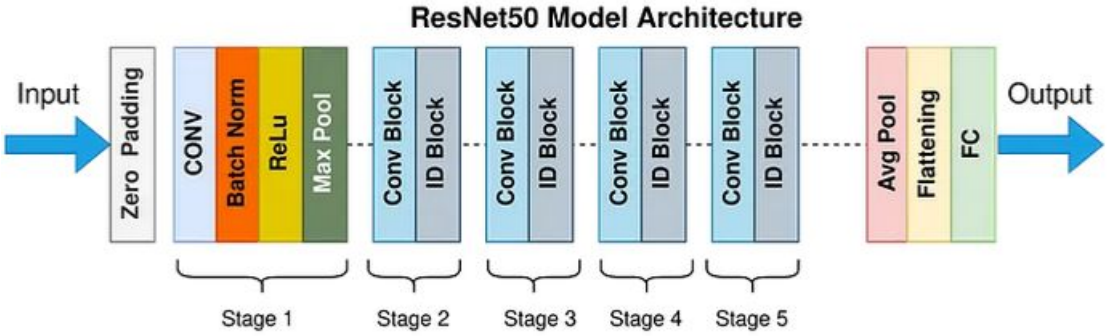


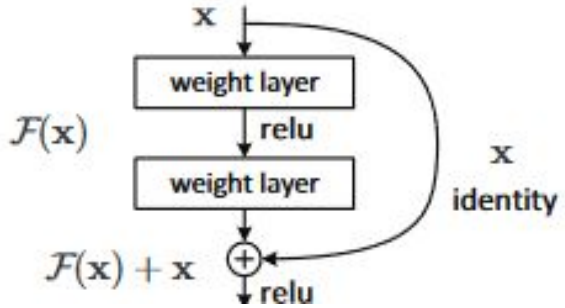
ResNet50

The remaining convolutional layers are grouped into four **stages**, each containing multiple **residual blocks**.

Each block has two or three convolutional layers with skip connections

Skip connections let information from previous layers to be preserved and reused in later layers, facilitating gradient flow and so improving the training process.

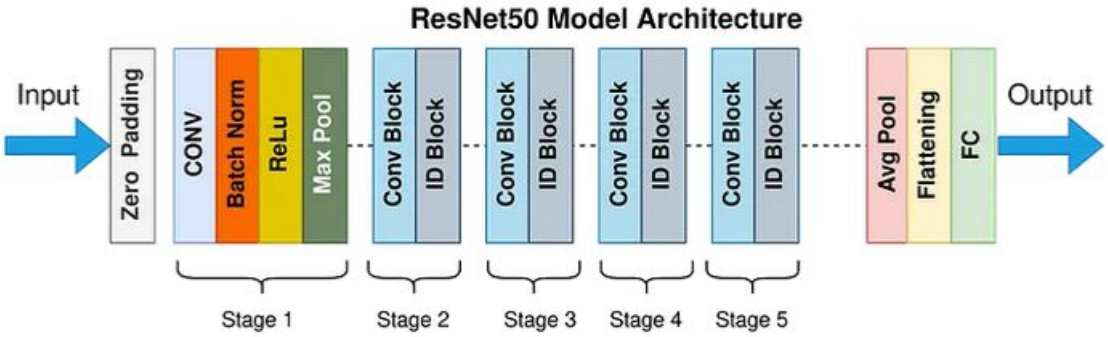


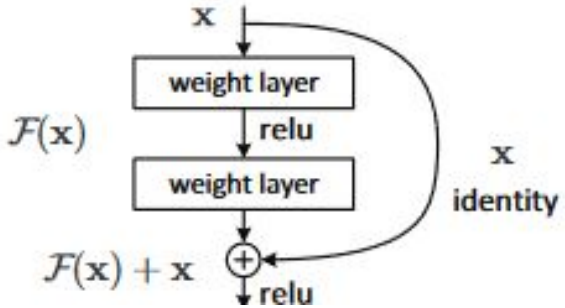


ResNet50

After the last stage a global average pooling layer and a fully connected layer are used to summarize the features and outputs the probability of the input image belonging to a specific class

For semantic segmentation to get pixel-wise predictions these layers are **removed**





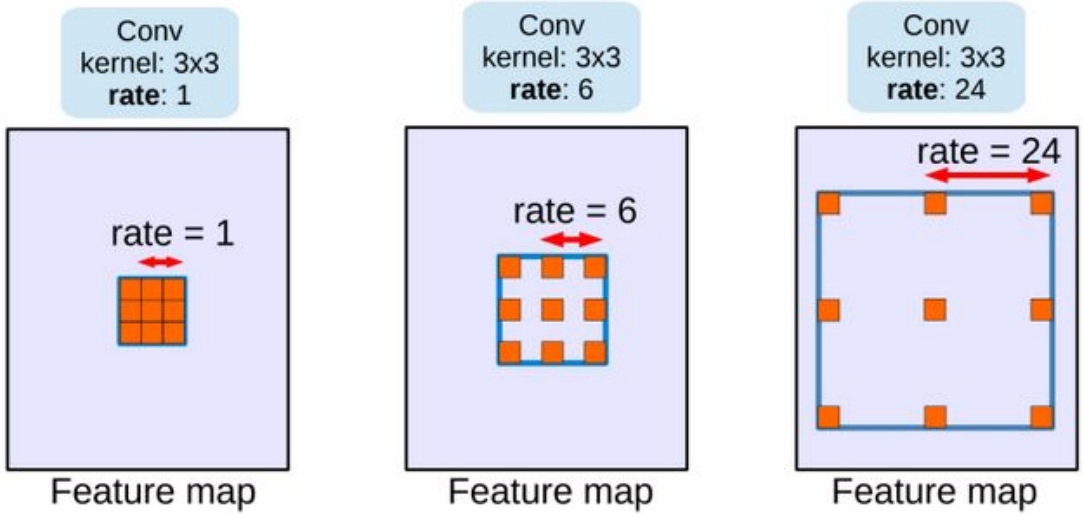
Atrous Convolution

In **standard convolution**, a filter is moved across an input image and applied at every location.

In **Atrous convolution** we apply the same filter filled with zeros between two consecutive filter values.

The number of zeros depend on the dilation rate.

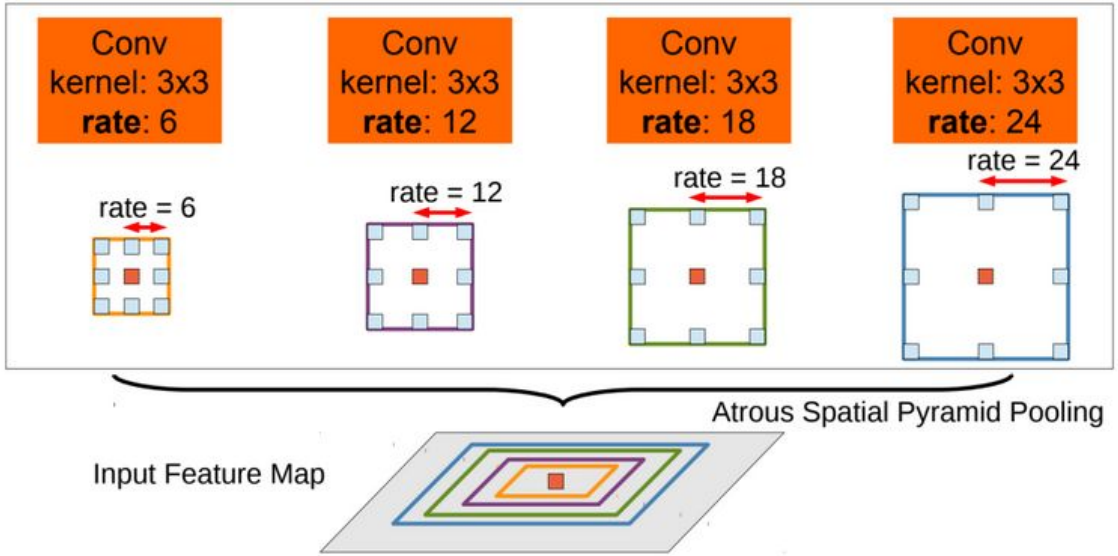
We obtain larger receptive fields without increasing the complexity of the filter, extracting features and informations from a broader area.



Atrous Spatial Pyramid Pooling

Atrous Spatial Pyramid Pooling (ASPP) is commonly used in semantic segmentation tasks to capture multi-scale contextual information.

ASPP takes in input feature maps generated from a previous CNN architecture, then applies parallel atrous convolutions with different dilatation rates to capture information at different scales.

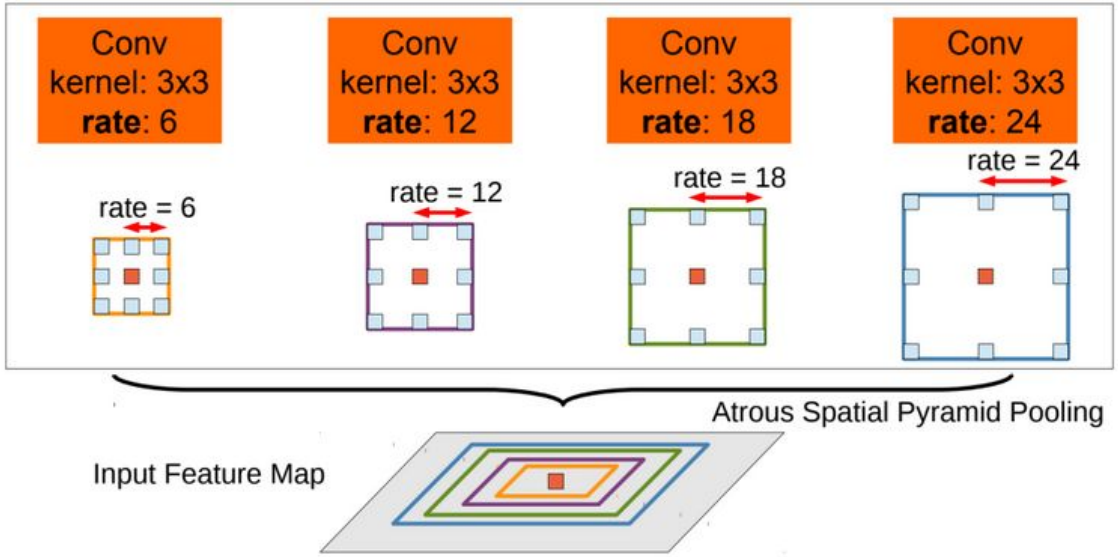


Atrous Spatial Pyramid Pooling

In parallel a global average pooling layer aggregates information across the input feature maps to capture global context.

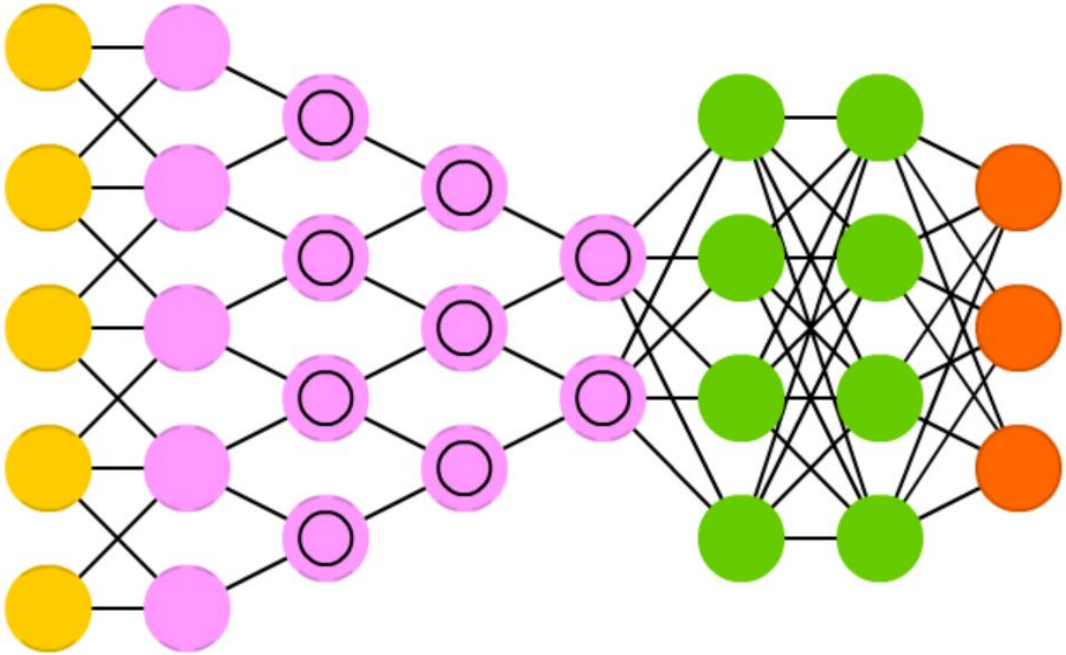
The results of the layers are upsampled to the original input dimension and concatenated, creating a multi-scale representation with local and global contextual information.

This representation can be further processed to reduce the dimensionality and extract more abstract features.



Models

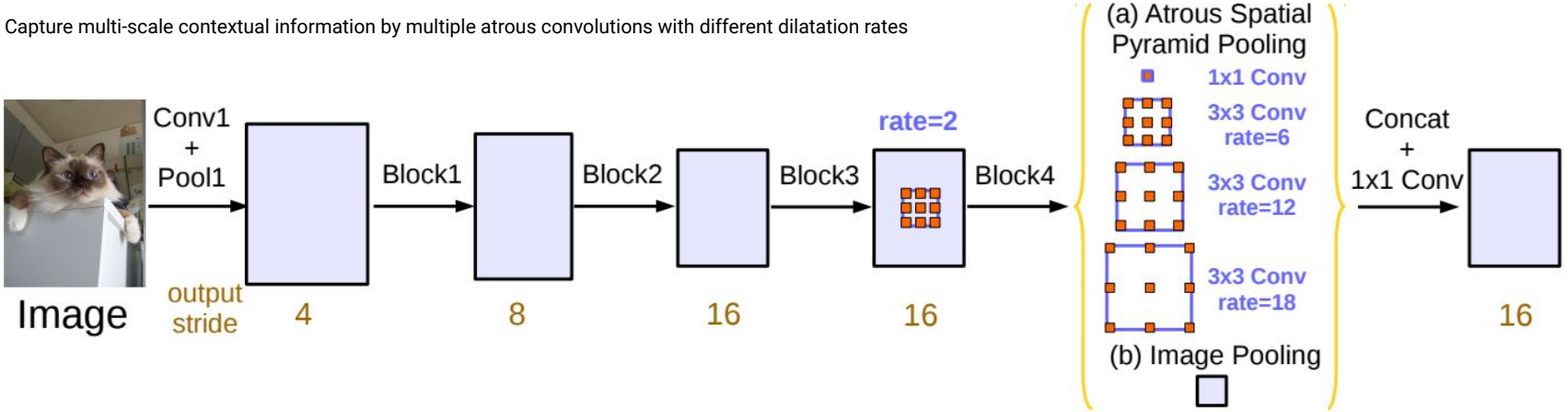
- DeepLabV3
- U-Net
- DenseASPP



Introduced by Chen et al. in Rethinking Atrous Convolution for Semantic Image Segmentation

DeepLabV3 - architecture

Capture multi-scale contextual information by multiple atrous convolutions with different dilatation rates



- **Backbone** blocks ResNet50 without the last Average Pooling and FC layer
- Feature maps are fed to an **ASPP** with different dilatation rates
- Output is upsampled to the size of the image to obtain the final segmented mask

DeepLabV3 - code

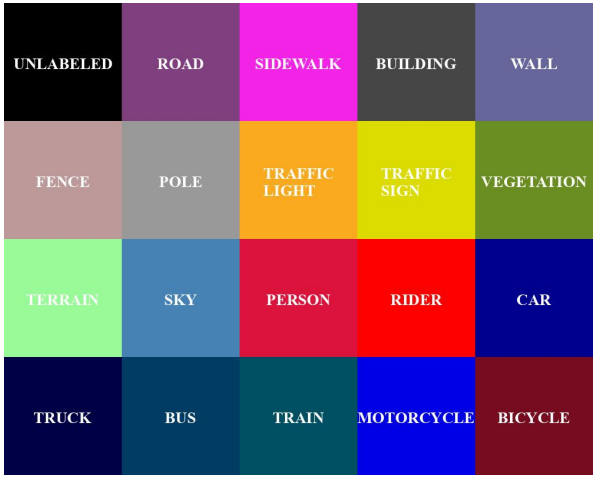
From **30** classes to **19** classes

```
    #       name                    id    trainId   category            catId     hasInstances   ignoreInEval   color
    Label(  'unlabeled'             ,  0 ,     255 , 'void'            , 0       , False        , True         , (  0,  0,  0) ),
    Label(  'ego vehicle'           ,  1 ,     255 , 'void'            , 0       , False        , True         , (  0,  0,  0) ),
    Label(  'rectification border'  ,  2 ,     255 , 'void'            , 0       , False        , True         , (  0,  0,  0) ),
    Label(  'out of roi'            ,  3 ,     255 , 'void'            , 0       , False        , True         , (  0,  0,  0) ),
    Label(  'static'                ,  4 ,     255 , 'void'            , 0       , False        , True         , (  0,  0,  0) ),
    Label(  'dynamic'               ,  5 ,     255 , 'void'            , 0       , False        , True         , (111, 74,  0) ),
    Label(  'ground'                ,  6 ,     255 , 'void'            , 0       , False        , True         , ( 81,  0, 81) ),
    Label(  'road'                  ,  7 ,       0 , 'flat'            , 1       , False        , False        , (128, 64,128) ),
    Label(  'sidewalk'              ,  8 ,       1 , 'flat'            , 1       , False        , False        , (244, 35,232) ),
    Label(  'parking'               ,  9 ,     255 , 'flat'            , 1       , False        , True         , (250,170,160) ),
    Label(  'rail track'            , 10 ,     255 , 'flat'            , 1       , False        , True         , (230,150,140) ),
    Label(  'building'              , 11 ,       2 , 'construction'    , 2       , False        , False        , ( 70, 70, 70) ),
    Label(  'wall'                  , 12 ,       3 , 'construction'    , 2       , False        , False        , (102,102,156) ),
    Label(  'fence'                 , 13 ,       4 , 'construction'    , 2       , False        , False        , (190,153,153) ),
    Label(  'guard rail'            , 14 ,     255 , 'construction'    , 2       , False        , True         , (180,165,180) ),
    Label(  'bridge'                , 15 ,     255 , 'construction'    , 2       , False        , True         , (150,100,100) ),
    Label(  'tunnel'                , 16 ,     255 , 'construction'    , 2       , False        , True         , (150,120, 90) ),
```

from https://github.com/mcordts/cityscapesScripts/blob/master/cityscapesscripts/helpers/labels.py

DeepLabV3 - code

The 19 classes



snippets from the code…

```python
def create_deeplabv3(num_classes):
    model = models.segmentation.deeplabv3_resnet50(pretrained=False)
    model.classifier[-1] = torch.nn.Conv2d(256, num_classes, kernel_size=(1, 1))
    return model
```

```python
# define transform
transform = transforms.Compose([
    transforms.Resize((512, 1024)),
    transforms.ToTensor()
    ])

# Define the dataloader
dataset = CustomDataset(image_dir, mask_dir
dataloader = torch.utils.data.DataLoader(da
```

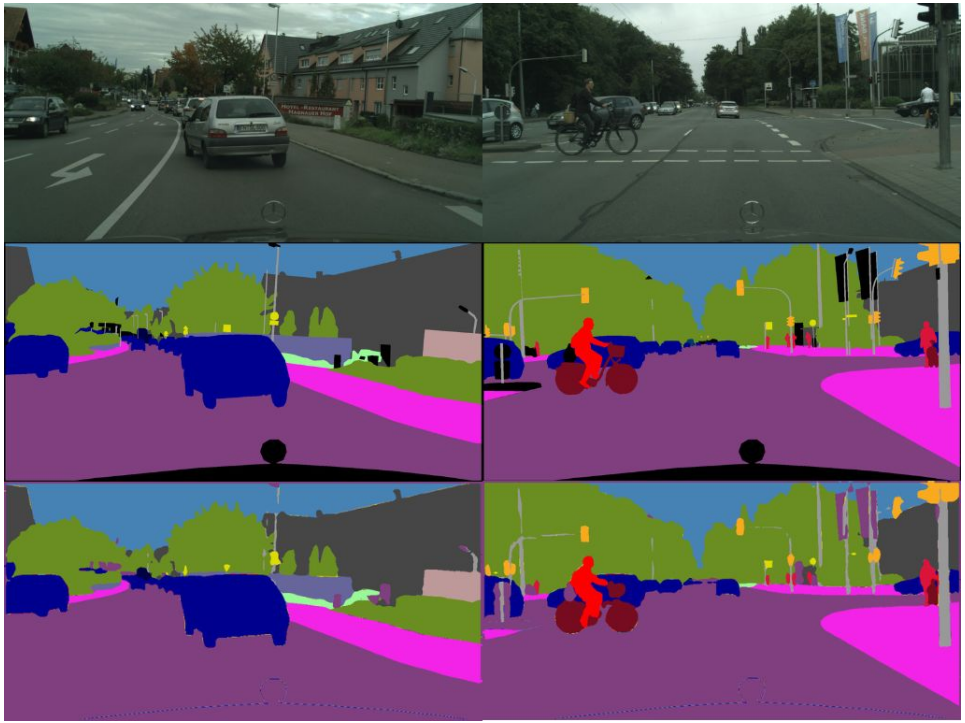
```python
class CustomDataset(Dataset):
    def __init__(self, image_dir, mask_dir, transform=None):
        self.image_dir = image_dir
        self.mask_dir = mask_dir
        self.transform = transform
        self.image_files = os.listdir(image_dir)

    def __len__(self):
        return len(self.image_files)

    def __getitem__(self, index):
        image_path = os.path.join(self.image_dir, self.image_f
        mask_path = os.path.join(self.mask_dir, self.image_fil

        image = Image.open(image_path).convert('RGB')
        mask = Image.open(mask_path)#.convert('L')
        # print(mask.getcolors())

        if self.transform is not None:
            image = self.transform(image)
            mask = self.transform(mask)
        return image, mask
```

DeepLabV3- results

Original Image

True Mask

DeepLabV3 Prediction

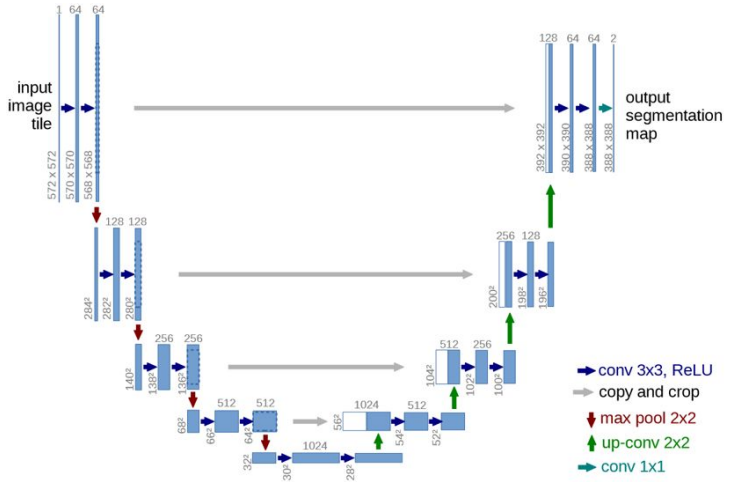


The 19 classes

UNLABELED	ROAD	SIDEWALK	BUILDING	WALL
FENCE	POLE	TRAFFIC LIGHT	TRAFFIC SIGN	VEGETATION
TERRAIN	SKY	PERSON	RIDER	CAR
TRUCK	BUS	TRAIN	MOTORCYCLE	BICYCLE

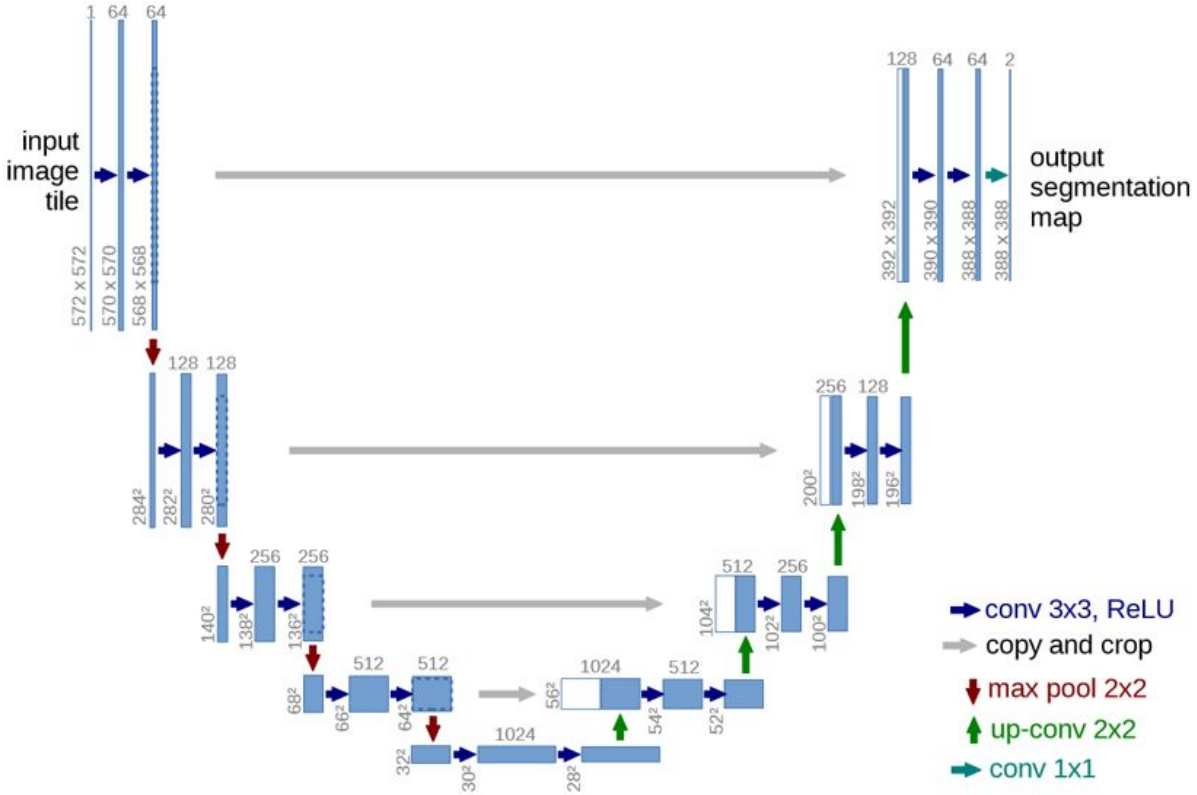
Unet - architecture

- developed for **biomedical** image segmentation at the Computer Science Department of the **University of Freiburg** in **2015**
- named after its **U-shaped design**, consisting of an encoder and a decoder
- widely used in various medical image analysis tasks, such as **segmenting organs, tumors**, or specific structures within medical scans



O.Ronneberger, P.Fisher, T.Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation
https://arxiv.org/abs/1505.04597

Unet - architecture



Unet - code

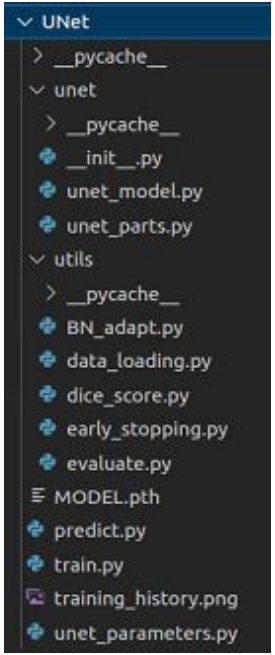
Original architecture credit:
https://github.com/milesial/Pytorch-UNet


```python
def unique_mask_values(idx, mask_dir, mask_suffix):
    mask_file = list(mask_dir.glob(idx + mask_suffix + '.*'))[0]
    colors = set()
    max_masks = 64
    for count, color in Image.open(mask_file).getcolors(max_masks): # if
        if isinstance(color,tuple): # RGB color
            colors.add(color)
        elif color in {0,255}: # binary color (black/white, False/True)
            colors.add(color == 255)
        else:
            raise AssertionError('this color palette for the masks is no
    colors = np.array( list(colors) ) # this is get the same output form
    return colors
```

with a LOT of modifications…

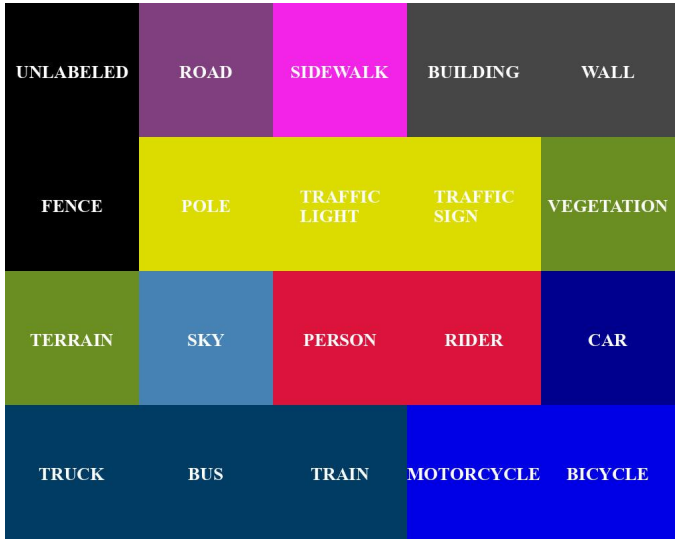
```python
# model parameters
epochs = 1000000
batch_size = 1
learning_rate = 1e-5
val_percent = 0.1
img_scale = 0.5
amp = True # use mixed precision (GradScaler)
bilinear = False # use bilinear interpolation for upsampling, https

# early stopping parameters. IMPORTANT: patience counter progressed
patience = 1000000
min_delta = 0

# optimizer parameters (RMSprop, with ReduceLROnPlateau scheduler)
weight_decay = 1e-8
momentum = 0.999
gradient_clipping = 1.0

# predict parameters
out_threshold = .5

# loss function: CrossEntropyLoss + Dice Loss
```

```python
class BasicDataset(Dataset):
    def __init__(self, images_dir: str, mask_dir: s
        self.images_dir = Path(images_dir)
        self.mask_dir = Path(mask_dir)
        assert 0 < scale <= 1, 'Scale must be betwe
        self.scale = scale
        self.mask_suffix = mask_suffix

        self.ids = [splitext(file)[0] for file in l
        if not self.ids:
            raise RuntimeError(f'No input file foun

        with Pool() as p:
            unique = list(tqdm(
                p.imap(partial(unique_mask_values,
                total=len(self.ids)
```



Unet - code

From **19** classes to **10** classes

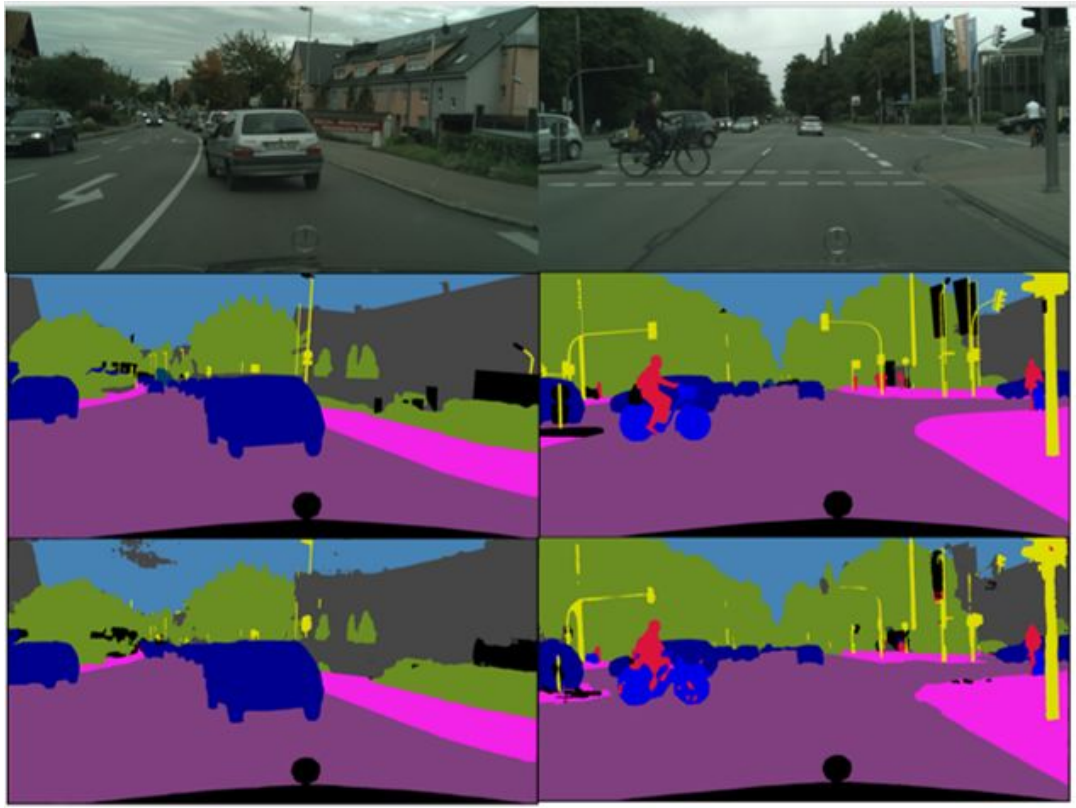


Unet - results

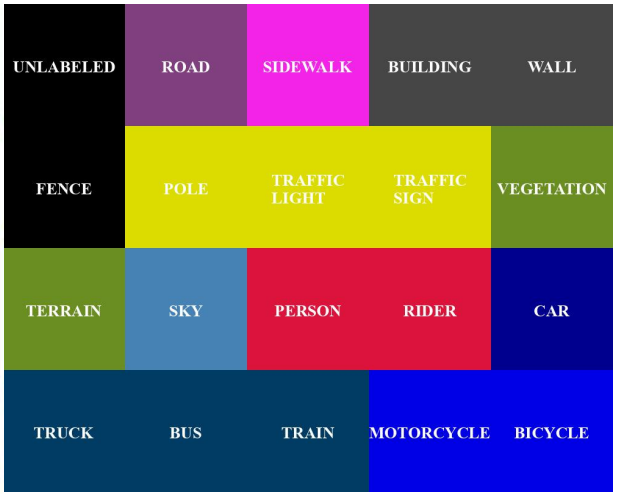
Original image

True mask

UNet prediction



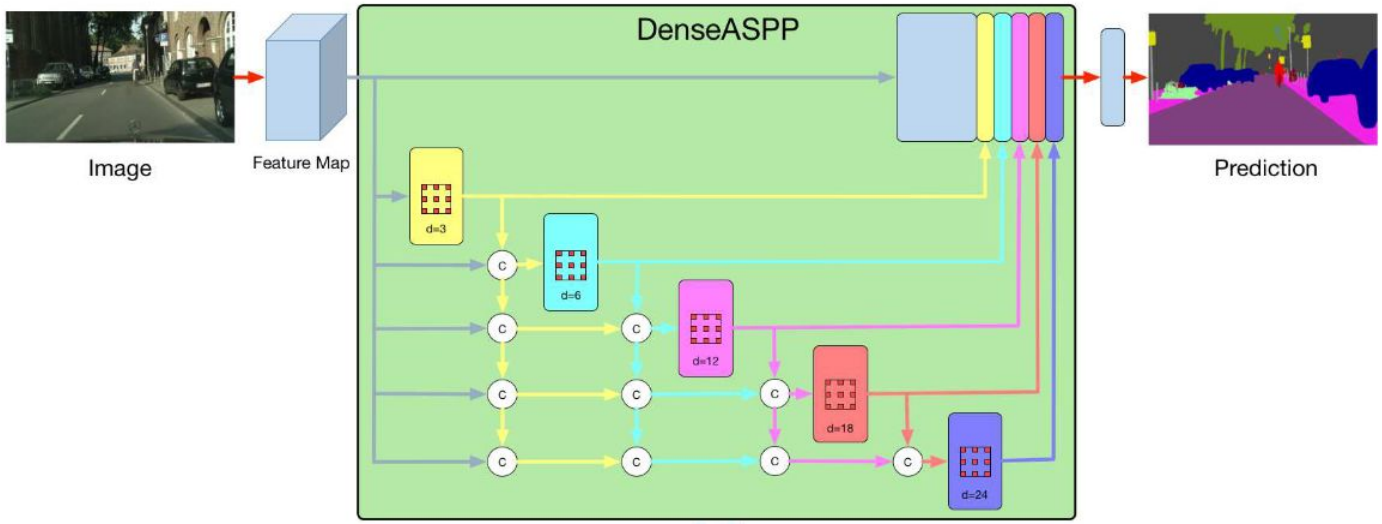
The 10 classes



DenseASPP - architecture

- DenseASPP architecture is based on the **ASPP** module and expands it with **dense connections**
- With dense connections the output of each atrous layer is concatenated with the input feature maps and all the outputs from previous layers
- Dense connections allow information to flow across different scales forming a **denser** feature pyramid, with better scale diversities, larger receptive field and more pixels involved in convolution than in ASPP
- DenseASPP improves the original ASPP module without increasing too much the overall complexity. This allows the model to produce more accurate and detailed semantic segmentation results.

DenseASPP - architecture



DenseASPP architecture: by dense connections each atrous layer receives in input the concatenation of the original feature maps and the outputs of the previous layers

DenseASPP – code

Original architecture and pretrained model credit:
https://github.com/DeepMotionAIResearch/DenseASPP

Made on: PyTorch 0.3, Python 3.7
We used: PyTorch 2.0, Python 3.10.10
It didn't work anymore but **we refurbished it**!

```
nn.init.kaiming_uniform_(m.weight.data)
```

```
weight = rename_weights(weight)
```

```
# function to correct the weights in the DenseASPP algorithm (changes in th
def rename_weights(weights):
    keywords =     ['norm.1','relu.1','conv.1','norm.2','relu.2','conv.2']
    new_keywords = ['norm_1','relu_1','conv_1','norm_2','relu_2','conv_2']
    new_weights = {}
    for key in weights:
        contains_keyword = False
        for i,keyword in enumerate(keywords):
            if keyword in key:
                contains_keyword = True
                tmp = key.split(keyword)
                new_key = tmp[0] + new_keywords[i] + tmp[1]
                new_weights[new_key] = weights[key]
        if not contains_keyword:
            new_weights[key] = weights[key]
    return new_weights
```

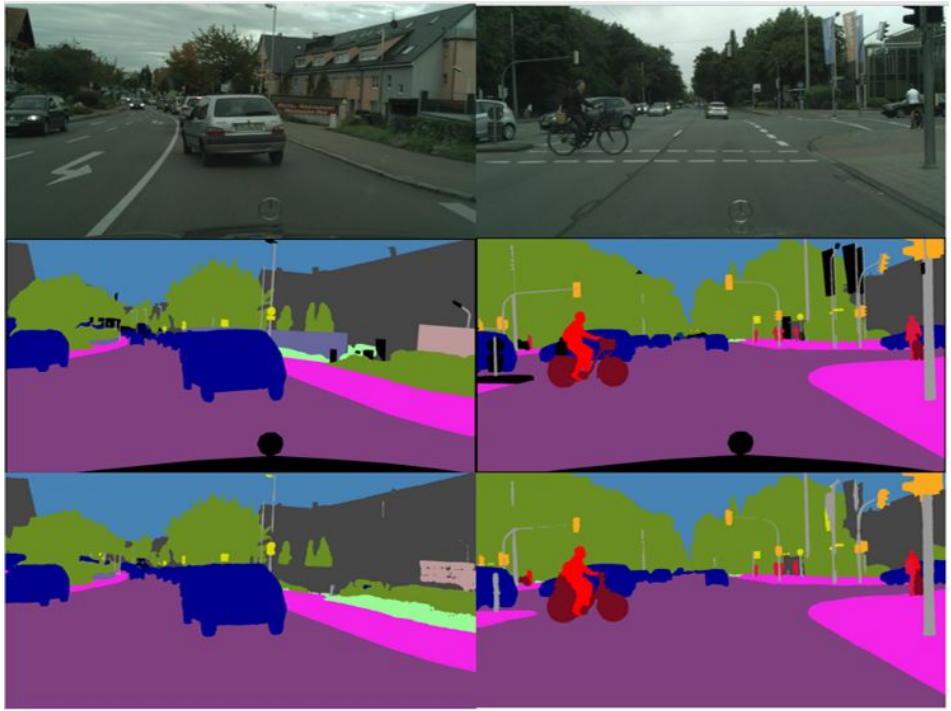
```
self.add_module('relu_1',
self.add_module('conv_1',

self.add_module('norm_2',
self.add_module('relu_2',
self.add_module('conv_2',
```

DenseASPP- results

Original image

True mask

DenseASPP prediction



The 19 classes

UNLABELED	ROAD	SIDEWALK	BUILDING	WALL
FENCE	POLE	TRAFFIC LIGHT	TRAFFIC SIGN	VEGETATION
TERRAIN	SKY	PERSON	RIDER	CAR
TRUCK	BUS	TRAIN	MOTORCYCLE	BICYCLE

Results

- Models comparison

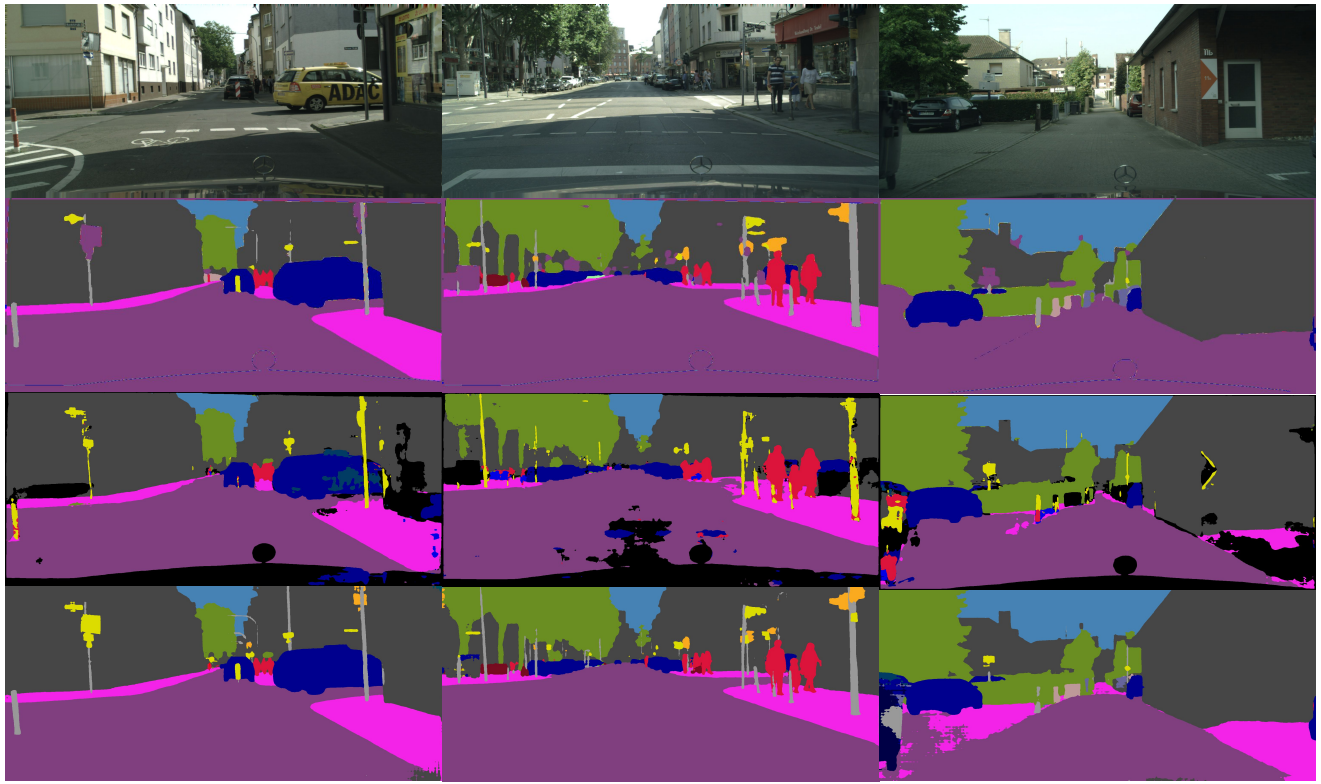
Results comparison

Original image

Deeplabv3

UNet

DenseASPP



Results tables

	DeepLabV3	Unet	DenseASPP
# classes	19	10	19
PerPixel	0.85	0.85	0.84
Micro	0.75	0.75	0.73
Macro	0.45	0.38	0.52
Weighted	0.75	0.78	0.74

We computed the **IOU** with the **Jaccard Similarity** function from sklearn.metrics. We have 3 possible ways to compute it:

- **average = 'Micro'** sums up the true positive, false positive, and false negative counts across all classes and then computes the Jaccard score.
- **average = 'Macro'**, also called **mean IoU**, calculates the Jaccard score separately for each class and then takes the average across all classes.
- **average = 'Weighted'** calculates the Jaccard score for each class and then takes the average, weighted by the number of true instances for each class.

We also coded a **per pixel** metric that counts the number of correct pixels and divides by the total number of pixels.

Results tables

	Truck	Car	Motorcycle	Bicycle	Bus	Train	Building	Rider	Person	Road	Wall	Sky	Vegetation	Sidewalk	Pole	Fence	Traffic Light	Traffic Sign	Terrain
DenseASPP	0.21	0.84	0.20	0.48	0.38	0.13	0.81	0.42	0.55	0.78	0.19	0.78	0.85	0.67	0.52	0.22	0.38	0.58	0.29
DeepLabV3	0.11	0.88	0.07	0.44	0.20	0.04	0.90	0.29	0.51	0.71	0.27	0.74	0.90	0.85	0.57	0.27	0.29	0.57	0.32

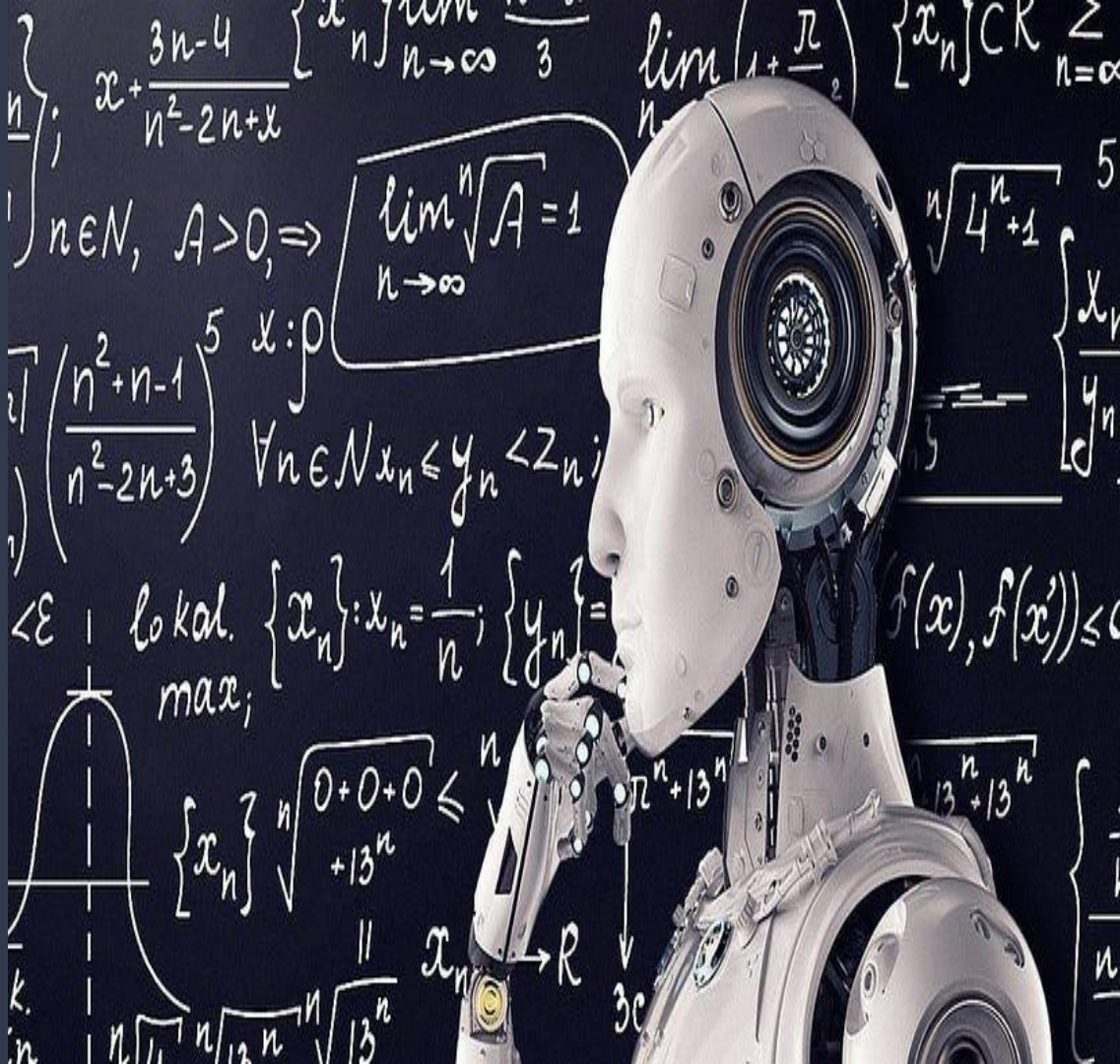
Why is the mean IoU so low?

This table shows the IOU for each possible different class.

It is due to 'Train', 'Truck', 'Motorcycle' and other uncommon classes which are **poorly identified**, potentially confused with similar classes such as ‘Car’, ‘Bus’, ‘Bicycle’

Conclusions

- Conclusions



Conclusions

Since the models have very similar performances for all the metrics except for **‘Macro’** we consider this as **discriminating**.

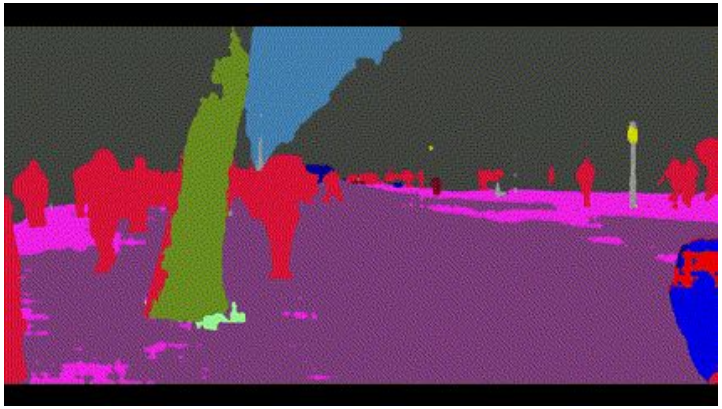
Despite the performances decline, it can be seen that DenseASPP performs better than Unet, while DeepLabV3 is in between.

The common feature with DeepLabV3 and DenseASPP is the use of the **ASPP**, so we can say that its use in itself is effective, but the addition of **dense connections** within the ASPP gives a significant improvement.

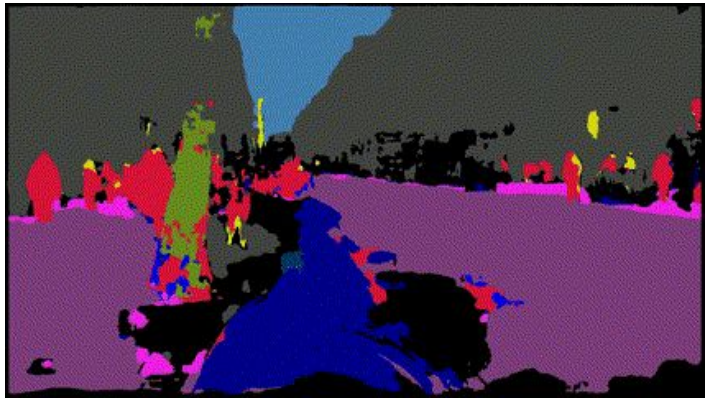
Video Segmentation

Original

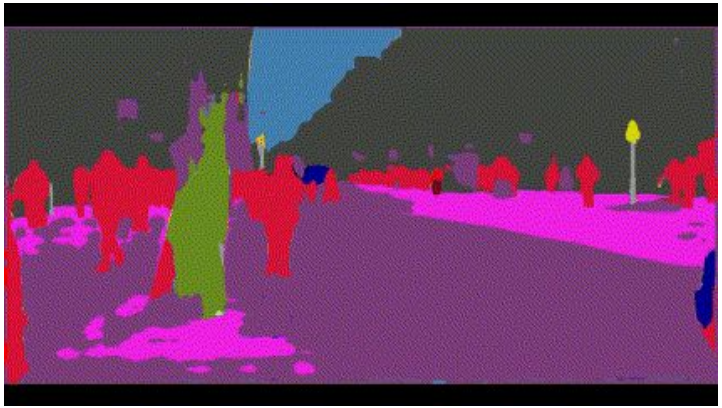
DeepLabV3



Unet



DenseASPP



Video Segmentation

Original

DeepLabV3

Unet

DenseASPP

Video Segmentation (Full Videos)

Original video
https://www.youtube.com/watch?v=NyLF8nHIquM&t=223s

DenseASPP
https://youtu.be/9YKO9olFXvA

UNet
https://youtu.be/FLVZmKb7txM

DeepLabv3
https://youtu.be/CxUpKj6oMFg