# Autoregressive Unsupervised Image Segmentation

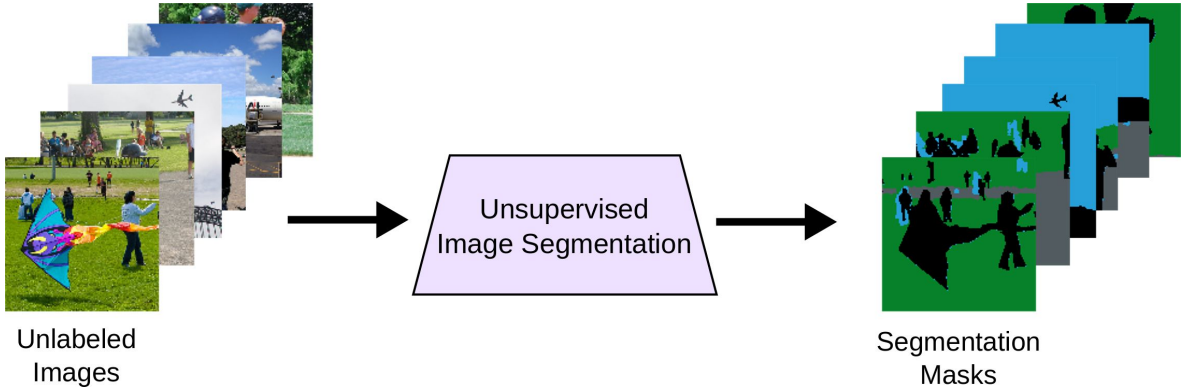
Yassine Ouali, Céline Hudelot and Myriam Tami

Université Paris-Saclay, CentraleSupélec, MICS, 91190, Gif-sur-Yvette, France

# Outline

# Overview

Objective: Unsupervised image segmentation, i.e. directly output segmentation maps without any supervision.

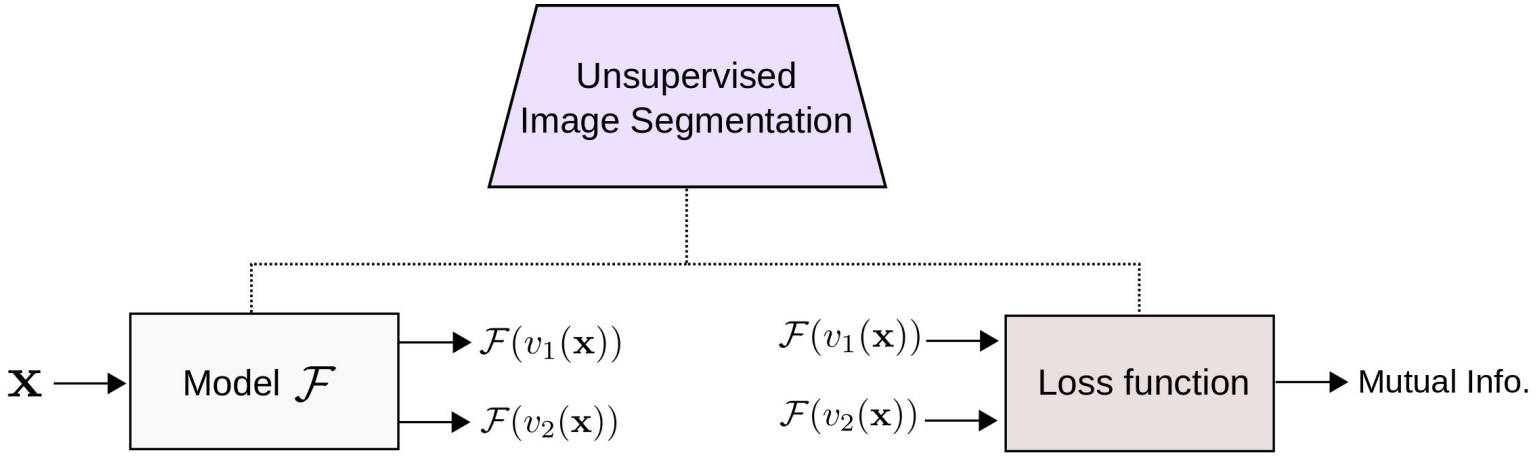



Why? To circumvent the expensive pixel-level annotation process.

# Overview

The proposed framework consists of two main components:

- An autoregressive model based on masked convolutions to construct difference views of the inputs.
- Mutual information based loss function as a learning objective.



# Autoregressive Generative Models

Likelihood-based models are capable of estimating $p_{\text{data}}$ from samples $x_1, \ldots, x_n \sim p_{\text{data}}$ and allow

- Computing p(x) for arbitrary x.
- Sampling x ~ p(x).

**Objective:** Estimate the true data distribution, i.e., represent the joint distributions over x with function approximators parametrized by θ, and learn θ so that $p_\theta(x) \approx p_{\text{data}}(x)$

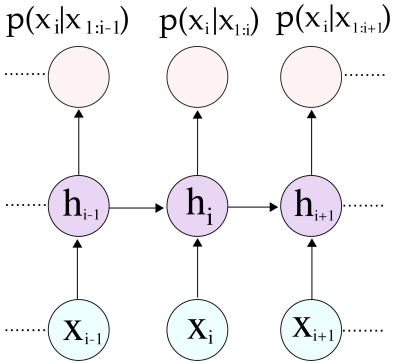
***Main question:*** *How to choose the right function approximators that are easy to train?*

**Autoregressive models:** Represent the joint distribution as a product of marginals.

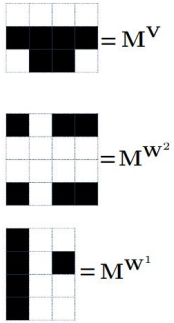
$$\log p(\mathbf{x}) = \sum_{i=1}^{d} \log p\left(x_i \mid \mathbf{x}_{1:i-1}\right)$$

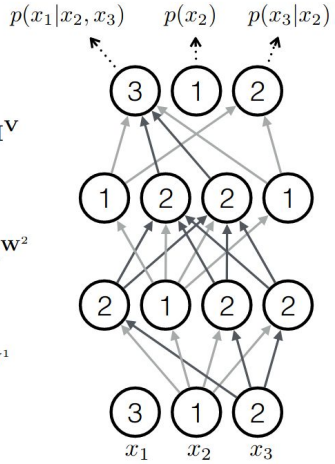
Such a formulation yields an expressive model for p(x) with tractable likelihood computation.

# Autoregressive Generative Models



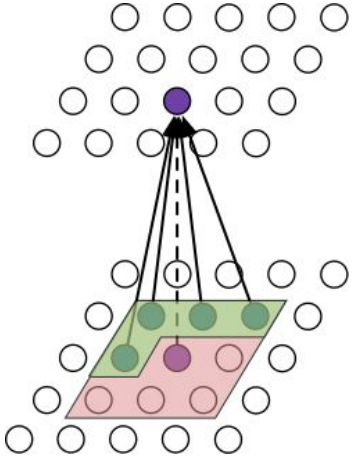

**RNNs**

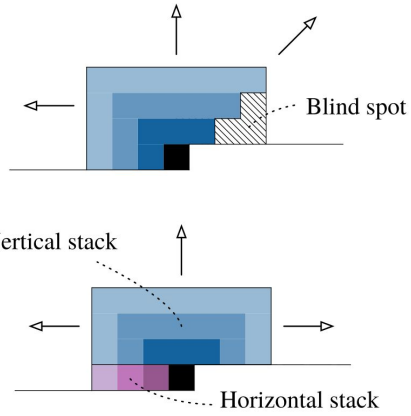



**MADE**

Mathieu et al. 2015.

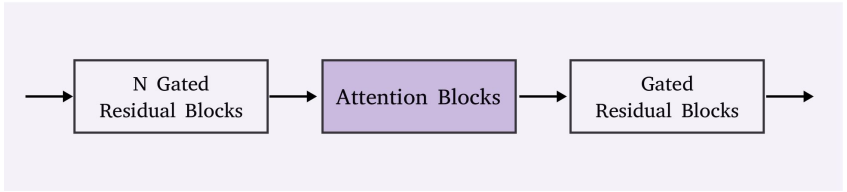





**Pixel CNN**

Van den Oord et al. 2016.

Snail Block

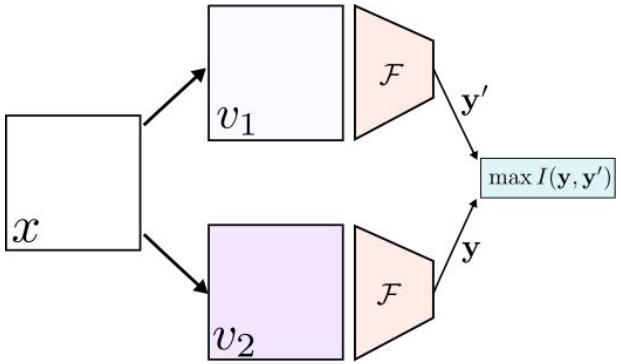



**Pixel SNAIL**

Chen et al. 2016

# Contrastive Learning & Mutual Information Maximization

Given two random variables v1 and v2 corresponding to two views of the input x, the objective is to learn a function to discriminate between *p(v1, v2)* and *p(v1)p(v2)*. i,e., Maximizing the Mutual Information *I(v1; v2)*.




Views v1 and v2 can be generated by:

- Data augmentation and cropping (MoCo *[He et al.]*, SimCLR *[Chen et al.]*, IIC *[Ji et al.]*).
- Image channels *(CMC [Tian et al.])*
- Spatial and temporal co-occurrences *(CPC [Oord et al.], CPCv2 [Hénaff et al.])*.
- The input and features at different scales (DeepInfoMax *[Hjelm et al.]*)

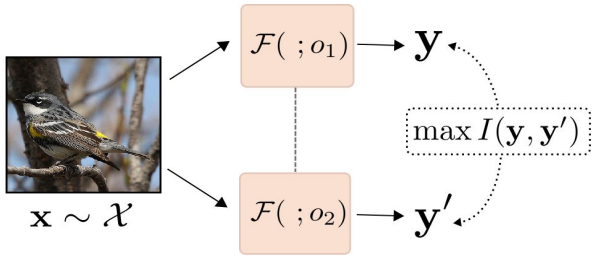
Maximizing a lower bound
for representation learning

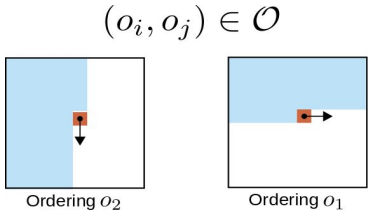
$$\max I(\mathbf{y}, \mathbf{y}') \rightarrow \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\log \frac{e^{f(\mathbf{y}_l, \mathbf{y}'_l)}}{\frac{1}{N}\sum_{m=1}^{N} e^{f(\mathbf{y}_l, \mathbf{y}'_m)}}\right]$$

Maximizing the extact value
for clustering

$$\max I(\mathbf{y}, \mathbf{y}') \rightarrow \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\mathbb{E}_{p(\mathbf{y},\mathbf{y}')} \log \frac{p(\mathbf{y}, \mathbf{y}')}{p(\mathbf{y})p(\mathbf{y}')}\right]$$

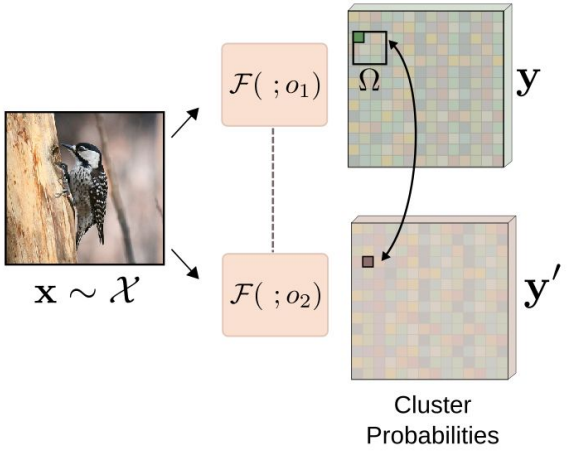
# Autoregressive Unsupervised Image Segmentation



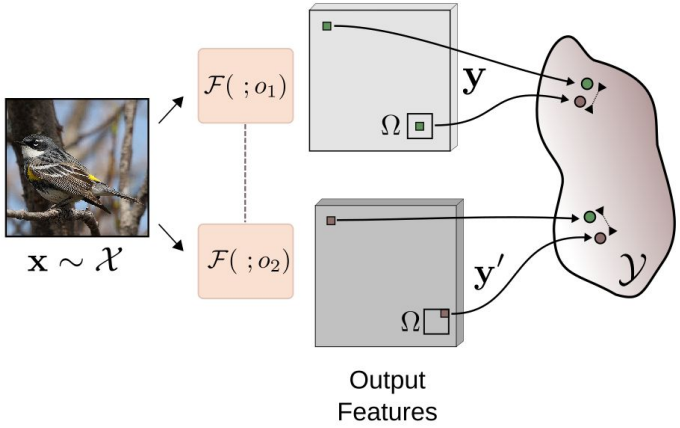

## Autoregressive Clustering (AC)

*Objective:* Similar cluster assignments



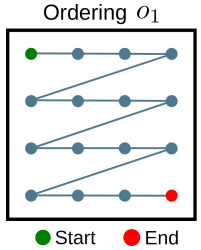

## Autoregressive Representation Learning (ARL)

*Objective:* Similar representations



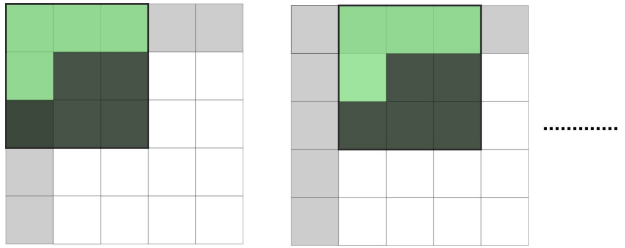
# Orderings as views

The normal ordering applied in Pixel CNN is a raster-scan ordering. Such ordering can be obtained by applying a simple masking to the weights of the convolution layers

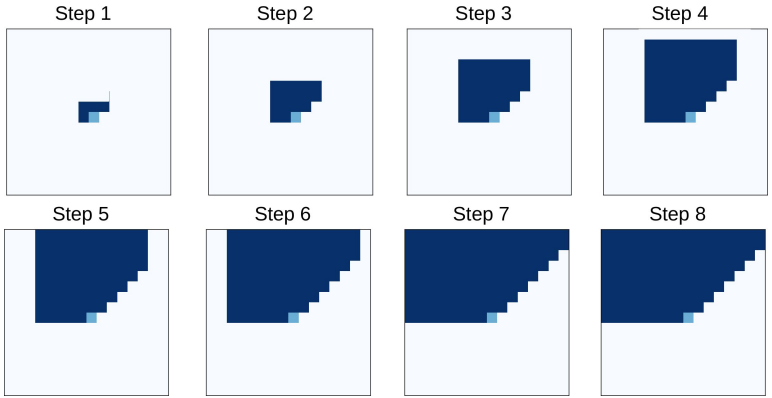



Conventional raster-scan ordering

Masked convolution



Application of masked convolution

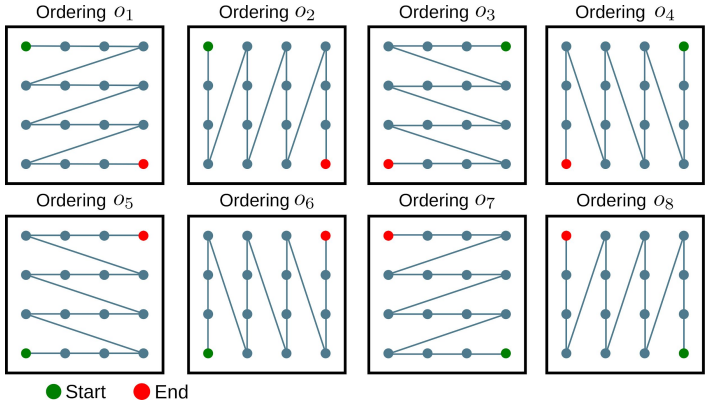



The resulting receptive field grows progressively

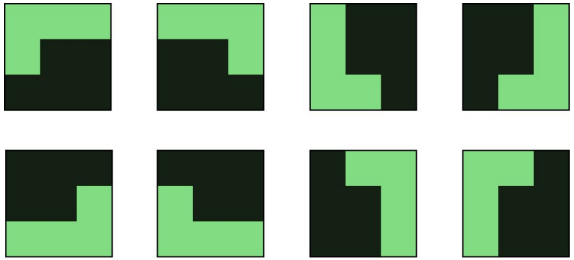
# Orderings as views

First, we start by increasing the number of possible orderings by using all 8 raster-scan type orderings. Such orderings could be obtained in a straightforward manner by simply masking the correct weights of the convolution layer.

**All 8 possible raster-scan type orderings**




**Masked convolutions**



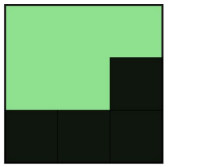
However, we can further reduce the number of masked weights, for more stable training and a faster growing of the receptive field.

# Orderings as views

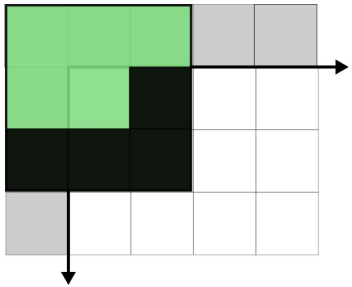
To further reduce the number of masked weights:

- We give access to the current pixel
- We apply a simple shift of the input with padding, we can avoid masking the first row of the weights without any change in the dependencies.

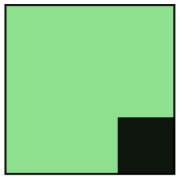
**Reducing the number of masked weights**



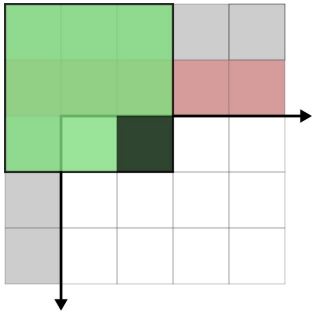
Access to the current pixel



Giving access to the current pixel



Simplified Version



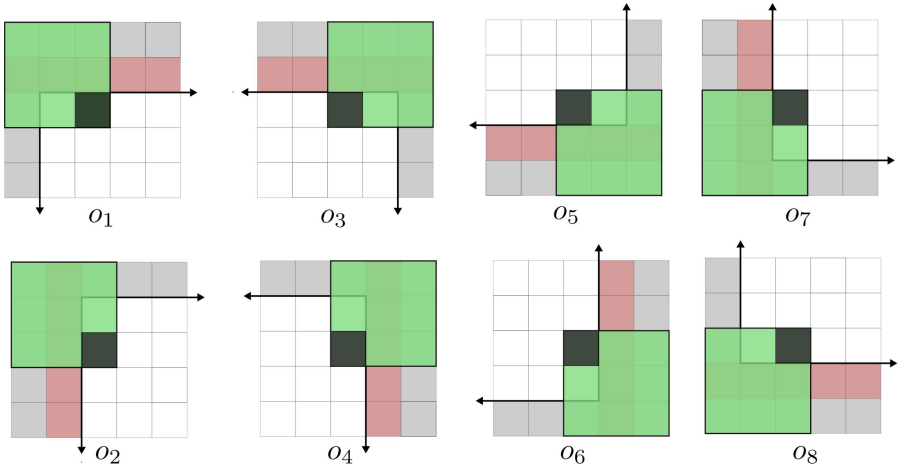
Further reducing the number of masked weights by a simple shit

# Orderings as views

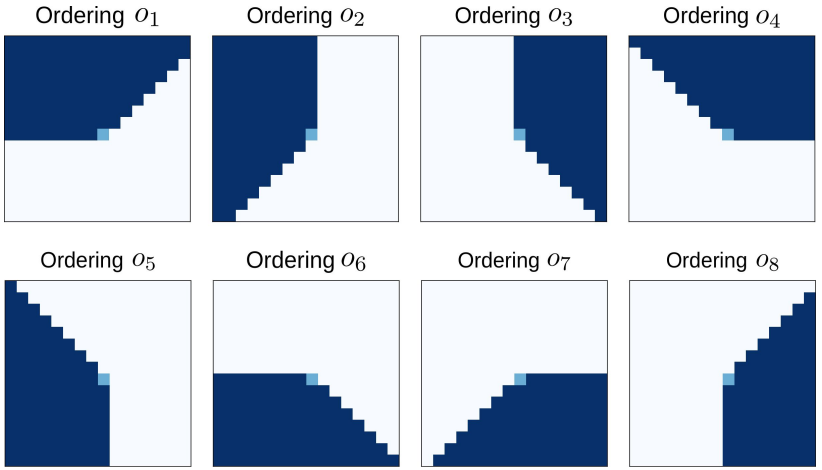
With four types of shifts and four types of masked convolutions, we can construct all of the 8 possible raster-scan type orderings.

The resulting receptive fields are the same as in the normal masking, but with less masked weights and a faster growing receptive field.

Masked convolutions with the corresponding padding

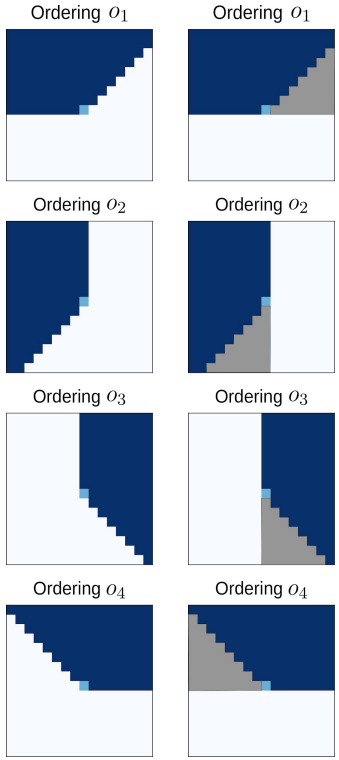



Resulting receptive fields



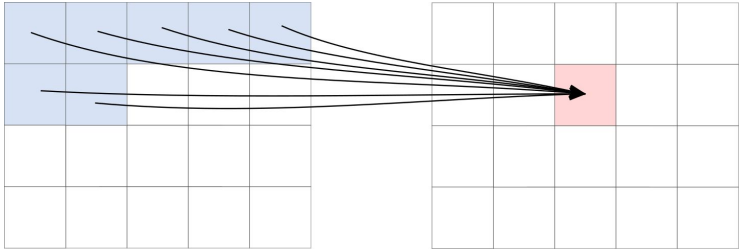
# Orderings as views

One drawback of current version of masked convolutions is their limited expressiveness since they create blind spots in the receptive field. This restricted receptive field can be overcome using self-attention.



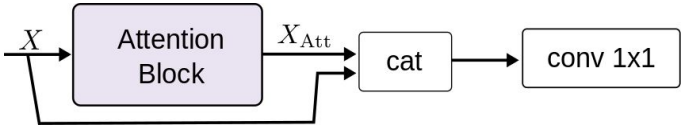

Blind spots

**Attention**



Each activation at given spatial location can directly fetch the relevant information from its dependencies

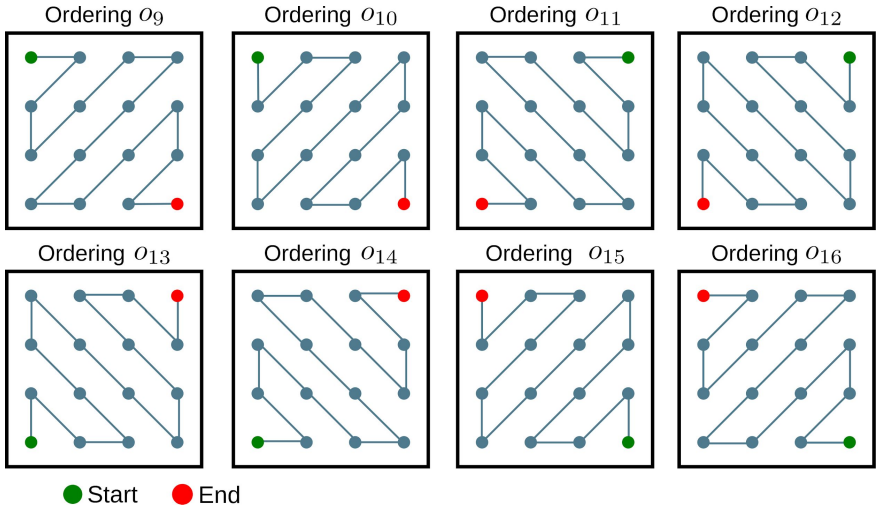
$$A = \text{Softmax}((QK^\top) \odot \mathbf{M}_{o_i})V$$
$$X_{\text{att}} = AW^O$$



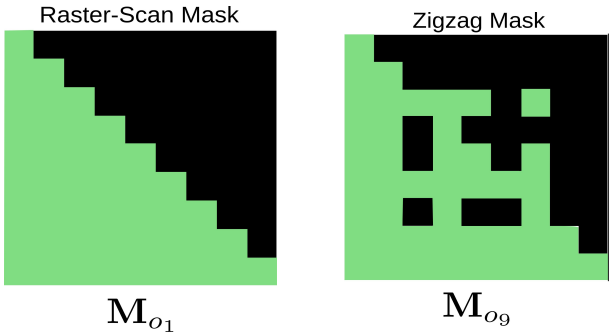
# Orderings as views

Using attention gives us another benefit, we can extend the set of possible orderings to include zigzag type orderings, by simply applying the correct attention masking.

## Zigzag type orderings




## Attention masks



# Training procedure

1- Sample two valid orderings $(o_i, o_j) \in \mathcal{O}$

2- Set the correct masking and padding for $o_i$

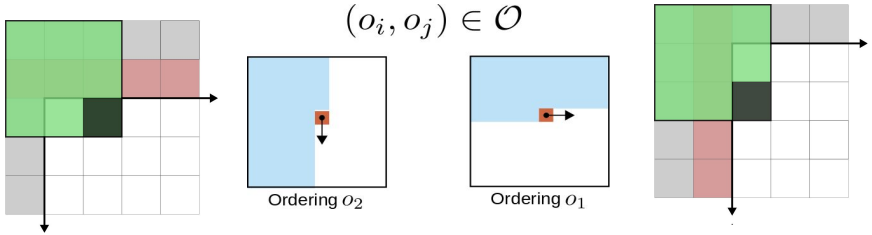
- Compute the output corresponding to the first view

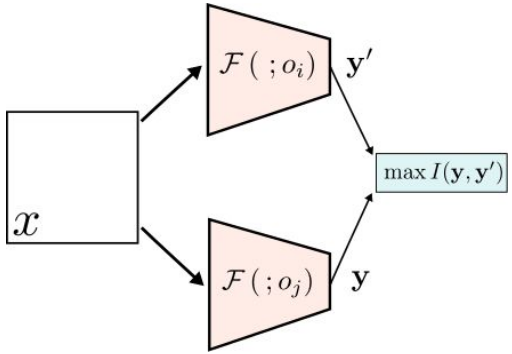
3- Set the correct masking and padding for $o_j$

- Compute the output corresponding to the second view

4- Compute the loss

5- Backpropagate and only update the unmasked weights, masked weights remain unchanged




Maximizing a lower bound for representation learning

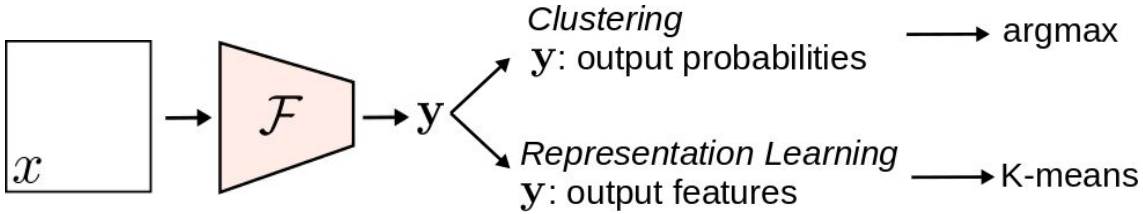
$$\mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\log \frac{e^{f(\mathbf{y}_l,\mathbf{y}'_l)}}{\frac{1}{N}\sum_{m=1}^{N} e^{f(\mathbf{y}_l,\mathbf{y}'_m)}}\right]$$

$\max I(\mathbf{y},\mathbf{y}')$

Maximizing the extact value for clustering

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\mathbb{E}_{p(\mathbf{y},\mathbf{y}')} \log \frac{p(\mathbf{y},\mathbf{y}')}{p(\mathbf{y})p(\mathbf{y}')}\right]$$

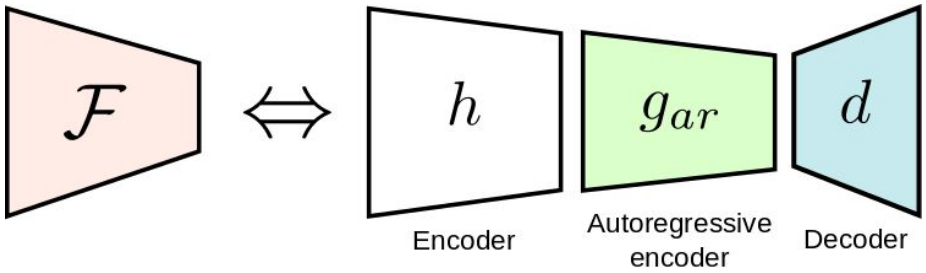
# Inference




During inference, we fallback to the normal convolution, where no masking or shifts are applied.

# Model

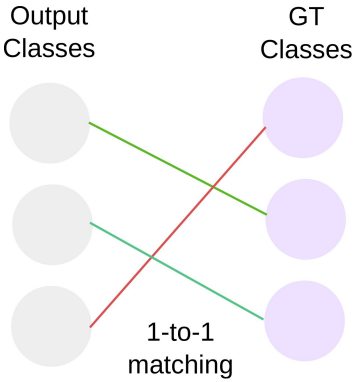
The model F can be implemented in an general manner using three subparts: $\mathcal{F} = h \circ g_{ar} \circ d$




- $h = id$, a fully autoregressive network,
- $g_{ar} = id$, normal encoder-decoder network,
- $h$ can be a simple conv-stem or a series of residual-blocks,
- $g_{ar}$ consists of masked convolutions & optional attention blocks.

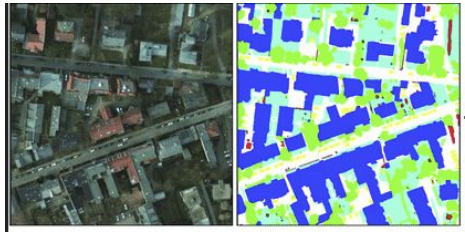
# Results

We report the accuracy obtained after conducting a one-to-one mapping.




**Datasets:** Potsdam with its two variants, with 6 and 3 classes, and COCO-Stuff, also with two variants.

**Potsdam**



6 classes

Roads and cars, vegetation and trees, buildings and clutter

3 classes

Merging each of the pairs

**COCO-Stuff dataset**

15 classes

3 classes

sky, ground and plants

# Results

### Variations of the model $\mathcal{F}$

| | Network $\mathcal{F} = h \circ g_{ar} \circ d$ $h$ | $g_{ar}$ | POS | POS3 |
|---|---|---|---|---|
| | Random | | 28.5 | 38.2 |
| $\mathcal{F}_1$ | Id | 5 Res. blocks | 39.3 | 56.3 |
| $\mathcal{F}_2$ | Stem | 5 Res. blocks | 46.4 | **66.4** |
| $\mathcal{F}_3$ | Res. block | 4 Res. blocks | **47.9** | 64.5 |
| $\mathcal{F}_4$ | 5 Res. blocks | Id | 35.1 | 63.4 |
| $\mathcal{F}_5$ | ResNet-18 | Id | 40.7 | 51.9 |

### Attention & type of orderings

| Orderings Raster-Scan | Zigzag | Attention | POS | POS3 |
|---|---|---|---|---|
| ✓ | × | × | 45.2 | 61.0 |
| ✓ | × | ✓ | 47.9 | 66.3 |
| × | ✓ | ✓ | 47.8 | **66.5** |
| ✓ | ✓ | ✓ | **49.3** | 65.4 |

### Dropout

| **p** | POS | POS3 |
|---|---|---|
| 0 | 46.4 | **66.4** |
| 0.1 | **47.9** | 64.7 |
| 0.2 | 46.9 | 65.1 |

### Number of oderings

| $\lvert\mathcal{O}\rvert$ | POS | POS3 |
|---|---|---|
| 2 | 43.2±2.19 | 59.5±5.12 |
| 4 | 45.6±3.22 | 63.55±3.52 |
| 8 | **46.4** | **66.4** |

### Sampling of the oderings

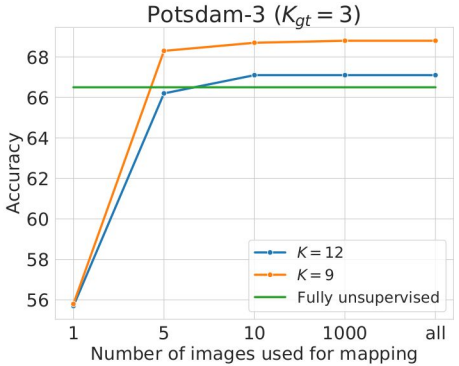
| Sampling $o_i$ | POS | POS3 |
|---|---|---|
| Random | 46.4 | **66.4** |
| No Rep. | 48.6 | 64.8 |
| Hard | **48.9** | 65.2 |

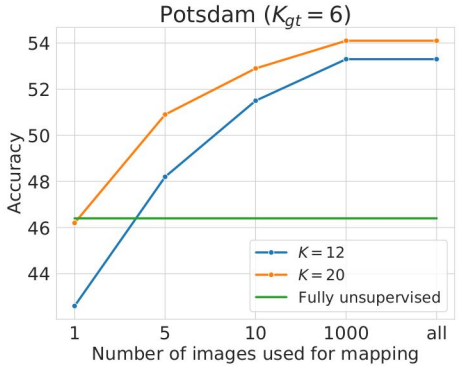
# Results

## Transformations

| Type | Transf. | POS | POS3 |
|---|---|---|---|
| None | - | 46.4 | 66.4 |
| Photometric | Col. Jittering | 47.9 | 65.5 |
| Geometric | Flip | 46.7 | 68.0 |
| Geometric | Rot. | **48.5** | 68.3 |
| Geo. & Pho. | All | **48.5** | **68.3** |

## Overclustering




## Autoregressive Clustering & Autoregressive Representation Learning

| Clustering | | |
|---|---|---|
| Method | POS | POS3 |
| Random CNN | 28.5 | 38.2 |
| AC | **46.4** | **66.4** |
| ARL | 45.1 | 57.1 |

| Linear Evaluation | | |
|---|---|---|
| Method | POS | POS3 |
| AC | **23.7** | **41.4** |
| ARL | **23.7** | 38.5 |

| Non-Linear Evaluation | | |
|---|---|---|
| Method | POS | POS3 |
| AC | **68.0** | **81.8** |
| ARL | 47.6 | 63.5 |

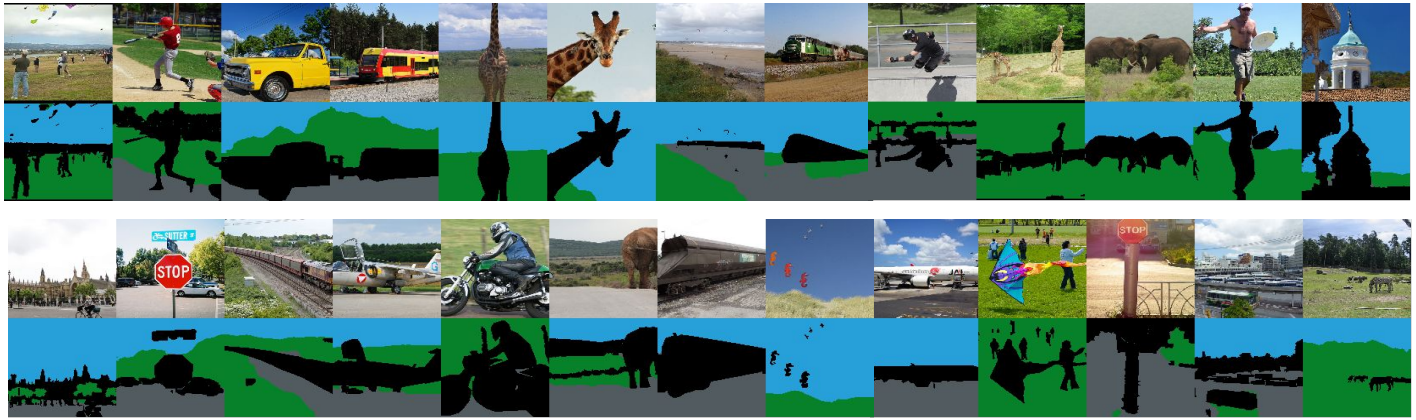
# Results

The proposed method also performs very competitively when compared to previous works, and capable of producing intricate segmentation.

## SOTA comparaison

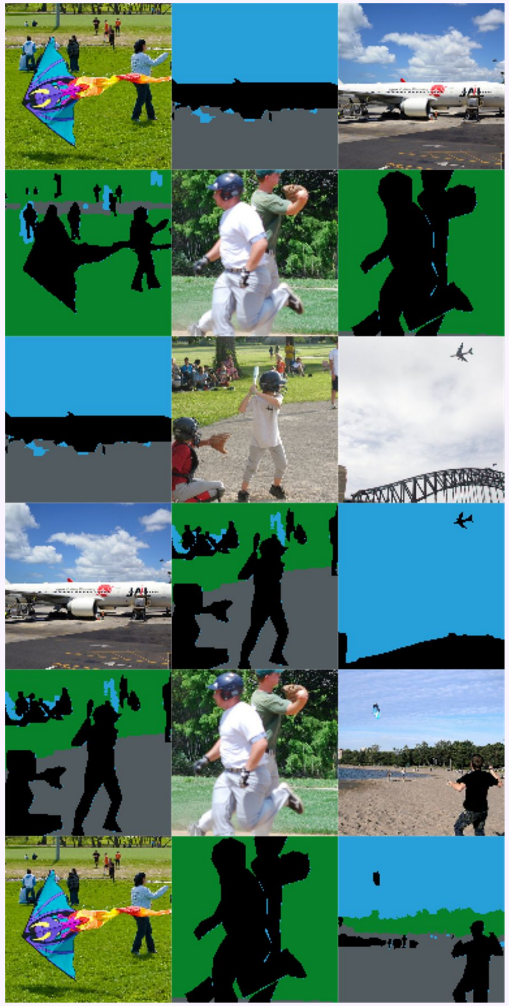
| | COCO-Stuff-3 | COCO-Stuff | Potsdam-3 | Potsdam |
|---|---|---|---|---|
| Random CNN | 37.3 | 19.4 | 38.2 | 28.3 |
| K-means [40] | 52.2 | 14.1 | 45.7 | 35.3 |
| SIFT [33] | 38.1 | 20.2 | 38.2 | 28.5 |
| Doersch 2015 [11] | 47.5 | 23.1 | 49.6 | 37.2 |
| Isola 2016 [26] | 54.0 | 24.3 | 63.9 | 44.9 |
| DeepCluster 2018 [6] | 41.6 | 19.9 | 41.7 | 29.2 |
| IIC 2019 [27] | 72.3 | 27.7 | 65.1 | 45.4 |
| AC | **72.9** | **30.8** | **66.5** | **49.3** |

## Qualitative Results



Yassine Ouali, Céline Hudelot and Myriam Tami

# Thank you



Project page: https://yassouali.github.io/autoreg_seg/

# References


- MADE: Masked Autoencoder for Distribution Estimation. 2015. Mathieu et al.
- Pixel Recurrent Neural Networks. 2016. Van den Oord et al.
- Conditional Image Generation with Pixel CNN Decoders. 2016. Van den Oord et al.
- PixelSNAIL: An Improved Autoregressive Generative Model. 2016. Chen et al.
- Contrastive Multiview Coding. 2019. Tian et al.
- Representation Learning with Contrastive Predictive Coding. 2018. Van den Oord et al.
- Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination. 2018. Wu el al.
- Data-Efficient Image Recognition with Contrastive Predictive Coding. 2019. Hénaff et al.
- Learning deep representations by mutual information estimation and maximization. 2018. Hjelm et al.
- Invariant Information Clustering for Unsupervised Image Classification and Segmentation. 2019. Ji et al.
- Learning Representations by Maximizing Mutual Information Across Views. 2019. Bachman et al.
- Momentum Contrast for Unsupervised Visual Representation Learning. 2020. He et al.
- A Simple Framework for Contrastive Learning of Visual Representations. 2020. Chen et al.