Self Supervised Learning

Theme

SSL can uncover important information present in training data, and encode similarities/ dissimilarities

Can we learn building blocks of causal graphical model by SSL?

How much supervision do we need?

Recent Work

- SSL with Data Augemntation (DA) provably isolates content from style [1]

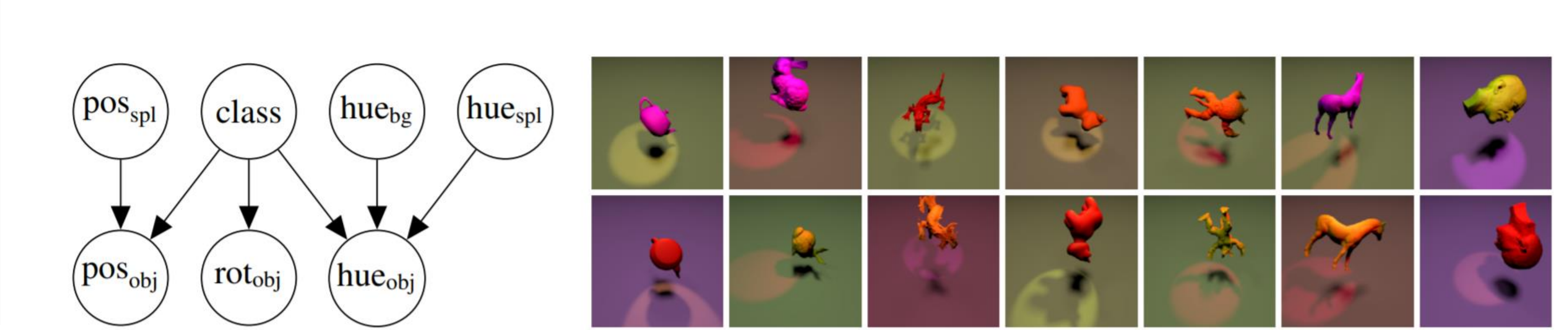
- Shows that SSL with learns a partition of the latent space into content vs style
- Style: latent codes which change when a data-point is transformed (rotate, color saturation, camera angle, resize,..)
- Content remains invariant
- Resembles a SCM where a CounterFactual is DA(x)
- Shows that we can uncover the causal structure from DA (to some degree)

[1] : (Kügelgen et al.) https://arxiv.org/pdf/2106.04619.pdf

Content vs Style

1. How to differentiate Content vs Style variables?

- It is implicitly expressed from the data augmentation methods we apply
- The partition can be learned from the information contained in dataset
- The method assumes that content variables are invariant in a DA
- Employs a ContrastiveLearning framework (InfoNCE objective) to leverage the similarity between original and augmented datapoint



[1] Figure 2: (Left) Causal graph for the Causal3DIdent dataset. (Right) Two samples from each object class.

Relation to Causal Factors

- Does identifying content variables in latent space always corresponds to causal factors?

a. The method shows good results in identifying content variables (invariant ones) from InfoNCE objective. By [1] Defn 4.1:

Definition 4.1 (Block-identifiability). We say that the true content partition $\mathbf{c} = \mathbf{f}^{-1}(\mathbf{x})_{1:n_c}$ is *block-identified* by a function $\mathbf{g} : \mathcal{X} \to \mathcal{Z}$ if the inferred content partition $\hat{\mathbf{c}} = \mathbf{g}(\mathbf{x})_{1:n_c}$ contains *all* and *only* information about $\mathbf{c}$, i.e., if there exists an *invertible* function $\mathbf{h} : \mathbb{R}^{n_c} \to \mathbb{R}^{n_c}$ s.t. $\hat{\mathbf{c}} = \mathbf{h}(\mathbf{c})$.

b. Used kernel ridge regression to predict ground truth c and s from the learnt representations $c^{\wedge} = g(x)$ and report $R^2 coefficient$. Highly accurate results on 3dIdnet dataset when there is a causal dependency between style and content variables

Generative process			R^2 (nonlinear)	
p(chg.)	Stat.	Cau.	Content c	Style s
1.0	✗	✗	**1.00** $\pm$ 0.00	0.07 $\pm$ 0.00
0.75	✗	✗	**1.00** $\pm$ 0.00	0.06 $\pm$ 0.05
0.75	✓	✗	**0.99** $\pm$ 0.00	0.02 $\pm$ 0.04
0.75	✓	✓	**0.96** $\pm$ 0.00	0.00 $\pm$ 0.00

[1] Page 8, R2 coefficients on 3dIdent dataset

Relation to Causal Factors

- Consider the image on right:
  - There is no recognizable object – a simple out-of-distribution sample
- Several other OOD examples: [link]
- **What if we attempt to learn the Content vs Style partition on a dataset of such images with their augmentations?**
  - Difficult to think what would be content or style
  - Perhaps no relation to learning ground truth latent causal factors
  - What information would an augmented pair help in encoding?

More with SSL

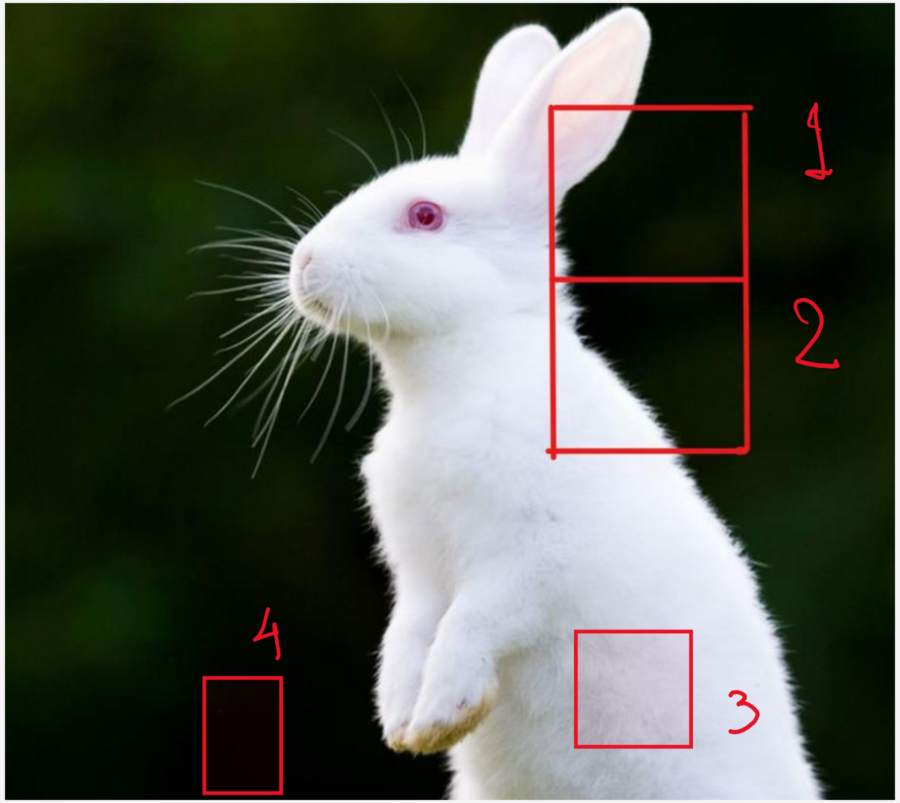
- My thoughts:
  - Contrastive Learning (CL) framework with data augmentation helps to uncover causal structure because common factors remain the same. The style attributes have “Content” variables as their Parents in SCM.
  - Along with DA, we can also utilize CL on patches of image.
    - What information can we get by pairing the two patches shown? For examples,
      - *With high probability, in this dataset,*

similarities →

an image can have sharp lines distinguishing two regions. A region can have a uniform solid color, or a tinge of multiple colors. (patch4 and patch3)

dissimilarities (counterfactuals) →

Boundary lines can be sharp (patch1) or diffused (patch2)



More with SSL

- Facts and counterfactuals can help to learn distinguishing features
- Learning positive correlations between convolutional patches within a sample image can bring more information
  - Consider the example on right (Fig1): the identity of these icons is *a mic/camera with a slash on top,* although the underlying white icon is not connected.
  - Pairing the conv patches 1 and 2 will give a high similarity score, indicating that they belong to same context.

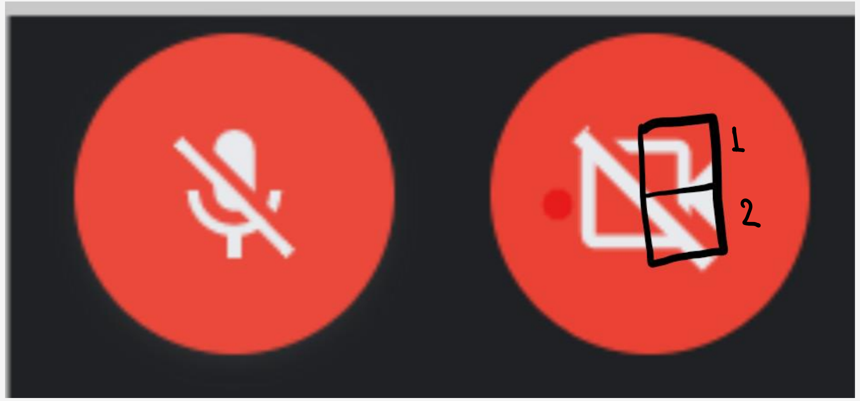


Fig 4.1: The icons for mic/camera off on Google Meet. Notice how we perceive it as a mic with a white slash on top, rather than two white scribbles separated by a red slant line

More with SSL

Likewise, we can learn the probability of continuation of line segment from such patches within the dataset. This falls in line with Gestalt Principle of continuation.

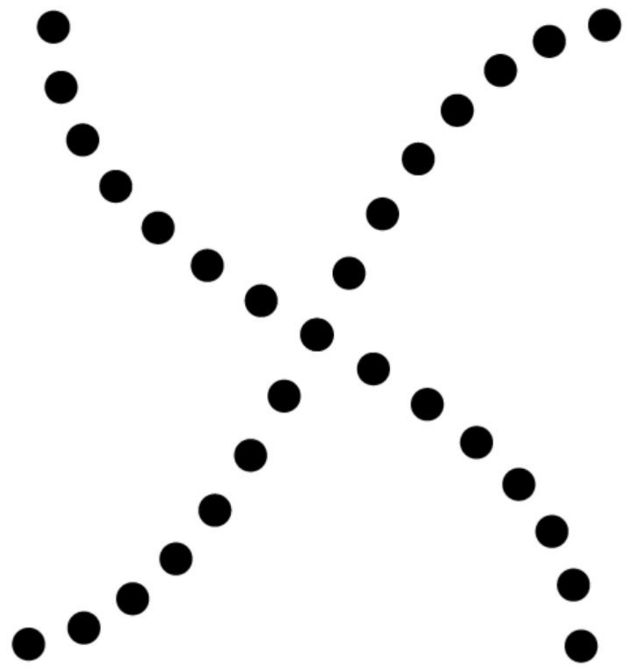


Figure 4. Good continuation would suggest that we are more likely to perceive this as two overlapping lines, rather than four lines meeting in the center.

Figure 5. Closure suggests that we will perceive a complete circle and rectangle rather than a series of segments.

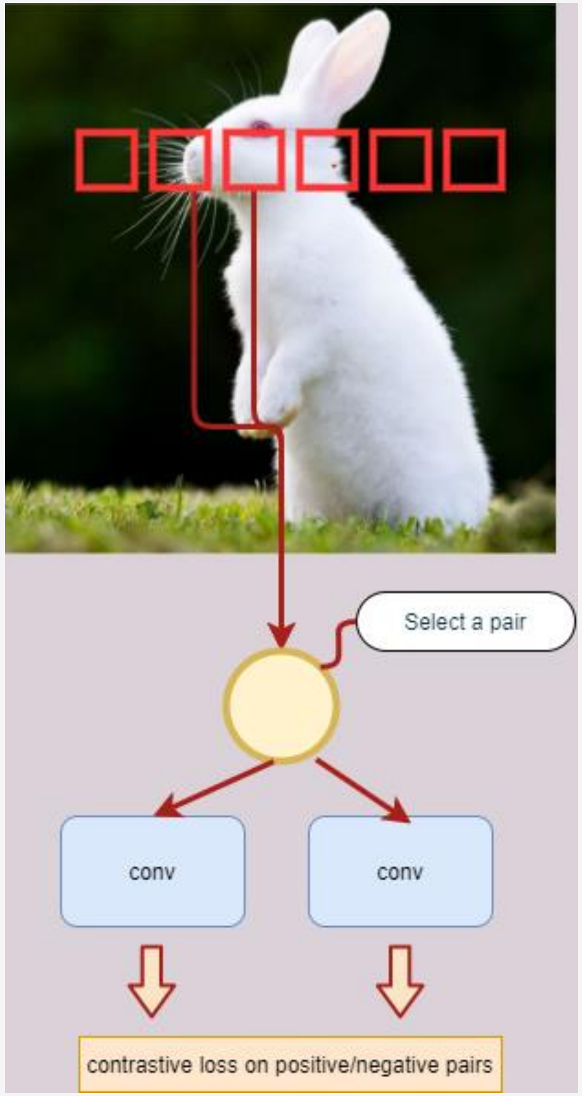
Img src: https://courses.lumenlearning.com/waymaker-psychology/chapter/gestalt-principles-of-perception/

How would this be useful?

- The way we can learn the probability of line segments being connected than disconnected, we can extract some information about various patches *without any supervision.*
- Key points:
  - Exploit similarities and dissimilarities between conv patch in dataset (images)
  - Repeat the above for different size of conv patches
  - Use already learned information to build a hierarchical model

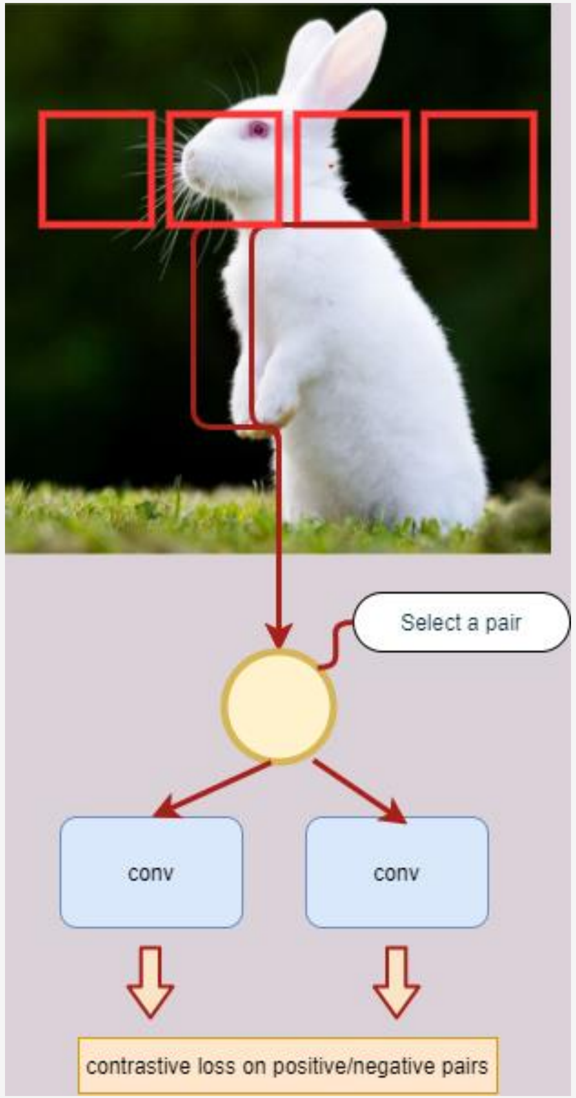
How would this be useful?

- Start with small window size
- Use a convNet to learn representations, followed by a CL loss
- How to decide a pair as positive/negative:
  - TBD
  - A simple threshold of distance between patch embeddings



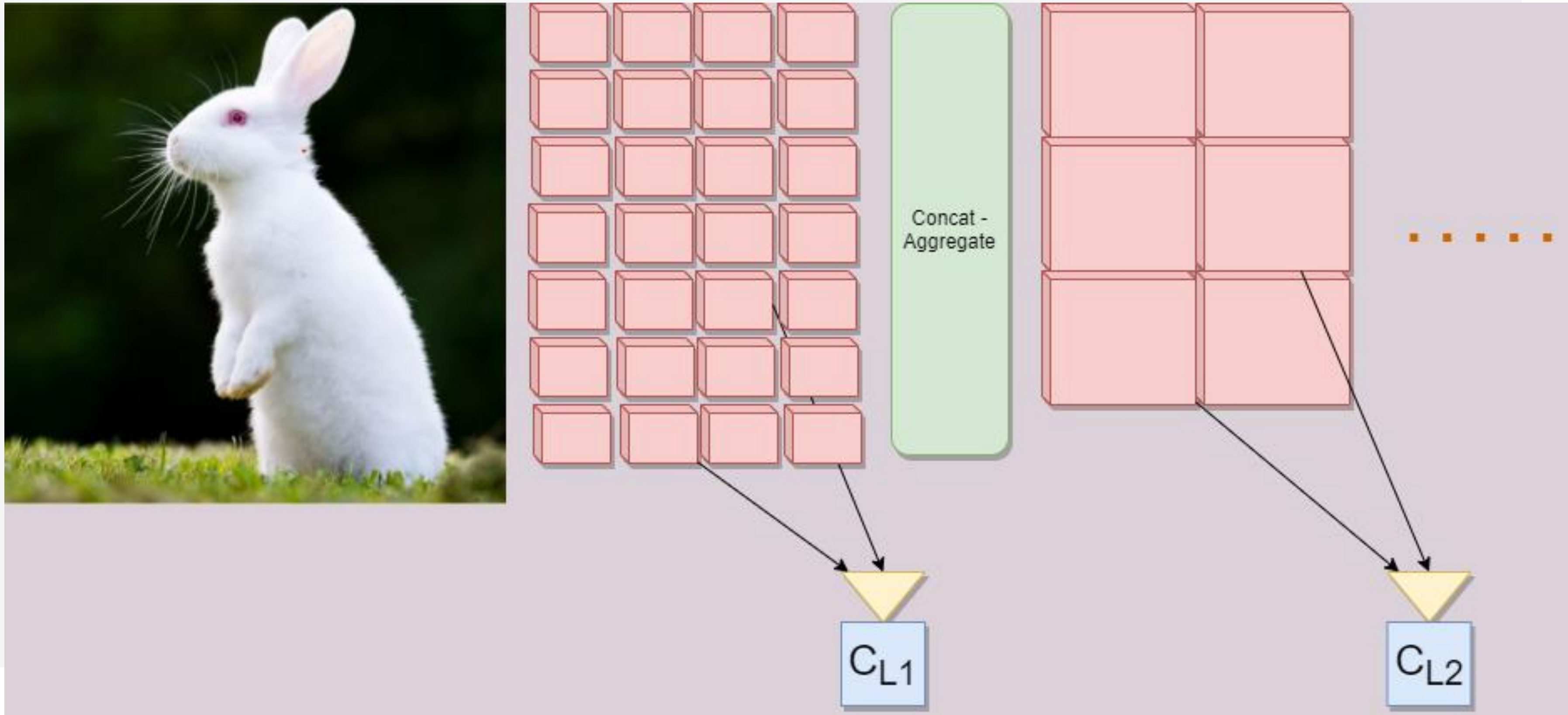
How would this be useful?

- Iterate with increasing window size
- The representation learnt in previous step should be used to perform this iteration
- Stack these steps?



How would this be useful?

- Joint learning:
  - At each step, create patches, choose a pair, pass through convNet, train on ContrastiveLoss (C_{Li}) objective
  - At step_K, concatenate embedding of constituent patches from step_K-1 to use information learned in the previous step
  - Minimize $Sum\ (C_{l1} + C_{l2} + ..)$



How would this be useful?

- Incremental learning:
    - At step_K, train only on C_{Lk} objective, and freeze conv layers till step_K-1. Train only the additional MLP while calculating C_L