Stacked Capsule AutoEncoders

Kosiorek AR, Sabour S, Teh YW, Hinton GE

Published Venue: NeurIPS 2019

Presenter: Sirnam Swetha

Faculty: Dr. Yogesh S Rawat



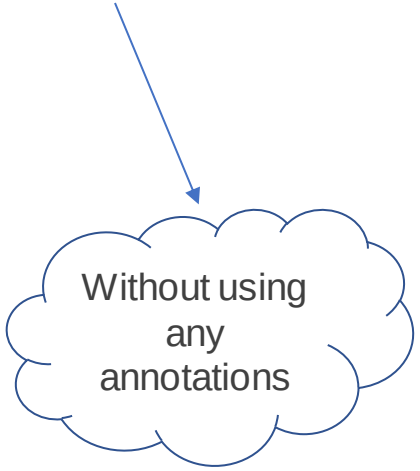
Contents

- Problem
- Motivation
- Preliminary Evaluation
- Overview
- Part Capsule Autoencoder
- Object Capsule Autoencoder
- Datasets
- Experiments
- Conclusion



Problem

Unsupervised Object Classification





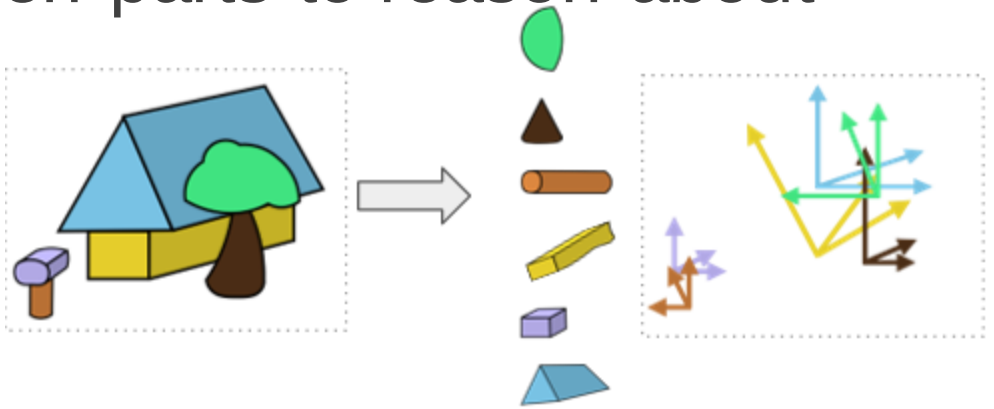
Cat

Related Works

- Representation Learning and clustering
  - Kosiorek et al.,[1] infer latent variables, one for each object in a scene
  - Burgess et al.,[2] perform unsupervised instance-level segmentation in an iterative fashion
- Using Mutual Information
  - Ji et al.,[3] maximizes an exact estimator of MI between two discrete probability vectors of input and its transformed image
  - DeepInfoMax[4] relies on negative samples and maximizes MI between the predicted probability vector and its input.
- Reconstruction
  - Transforming autoencoder[5] can reconstruct an affine-transformed version of the original image.

Motivation

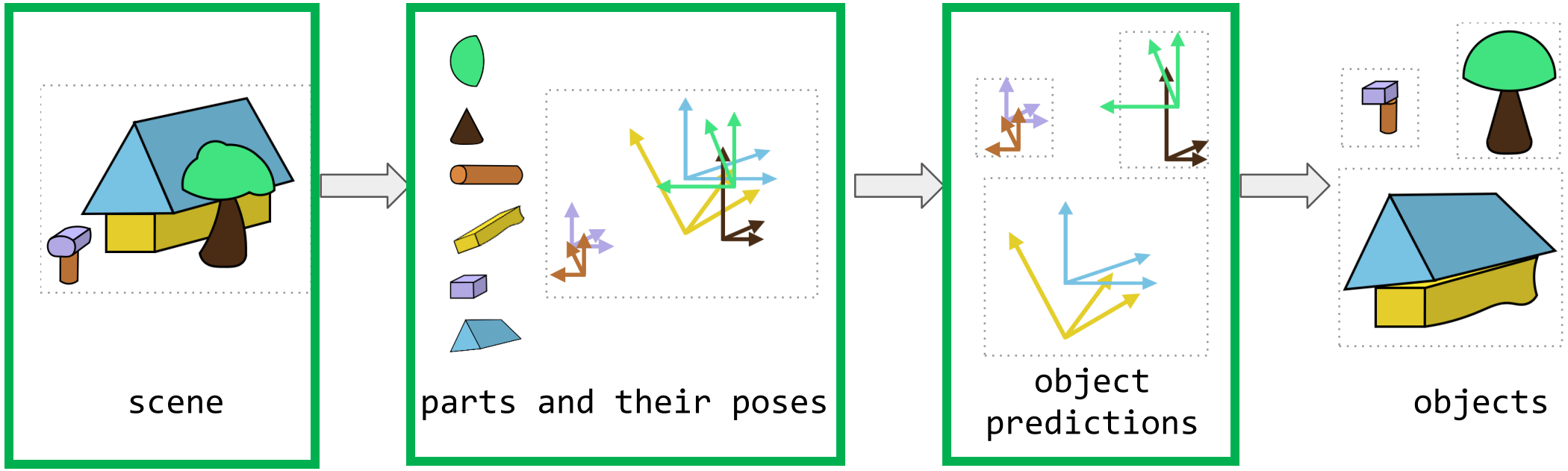
- Modern methods achieve great performance in object recognition, but demand staggering amounts of data.
- Decompose objects into their constituent parts
- Provide information about shape
- Model geometric relationships between parts to reason about objects.
- Better viewpoint generalization





Credits: http://akosiorek.github.io/ml/2019/06/23/stacked_capsule_autoencoders.html

Overview



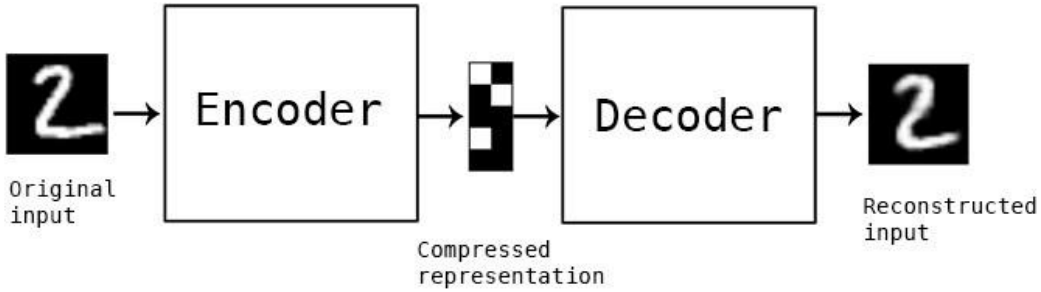
Capsule networks work by inferring parts & their poses from an image, and then using parts and poses to reason about objects.

Background
and
Proof Of Concept



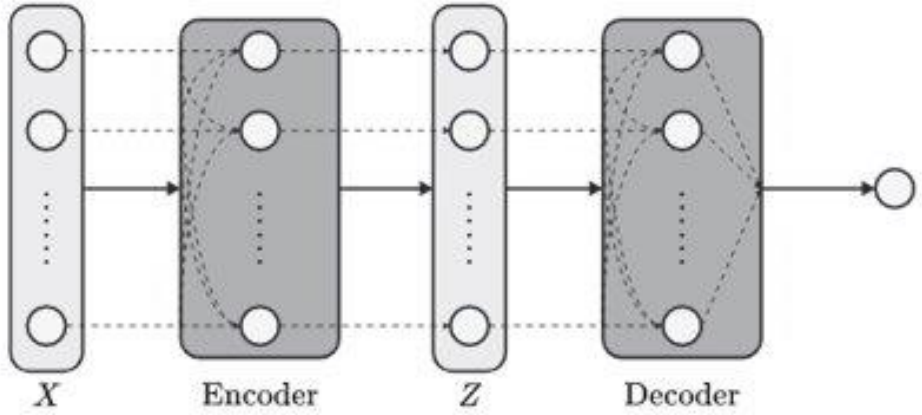
Background

Autoencoder



Autoencoder is an unsupervised artificial neural network that learns how to efficiently *compress* and encode data then learns how to reconstruct the data back.

Set Transformer

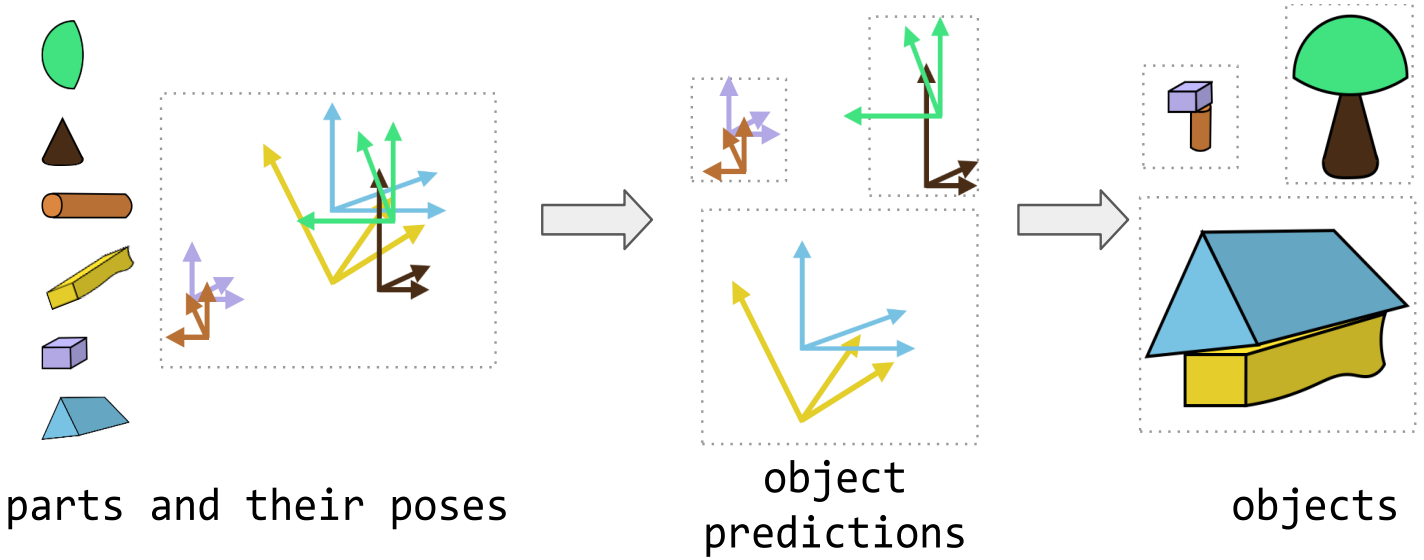


The model consists of an encoder and a decoder, both of which rely on attention mechanisms, specifically designed to model interactions among elements (being *permutation invariant*) in the input set.

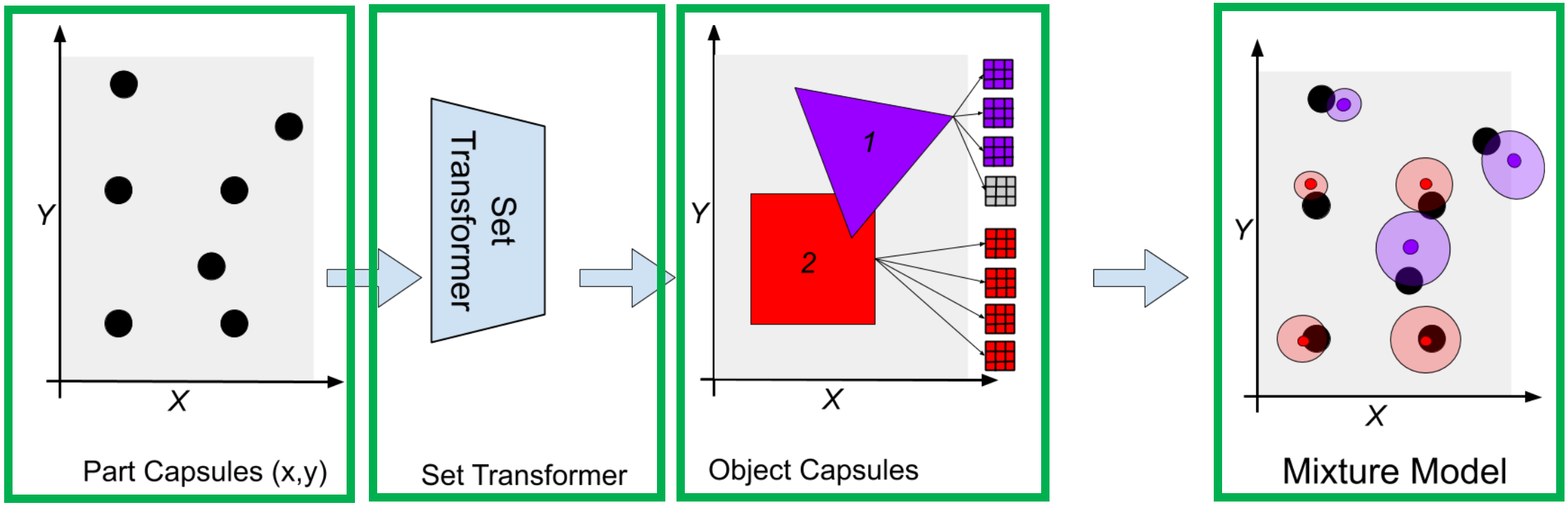
Preliminary Evaluation

Constellation Autoencoder as Proof Of Concept:

- Parts exhibit less variety in their appearance and shape than full objects



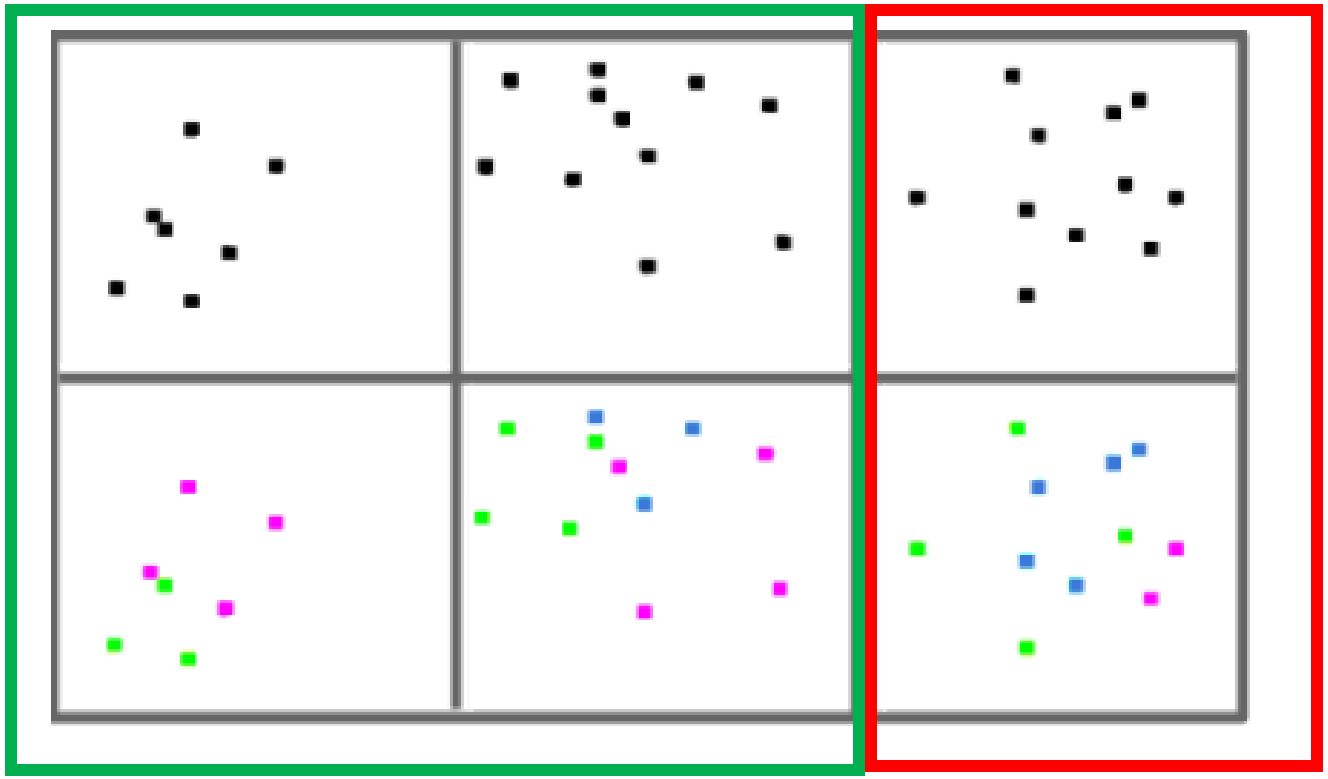
Constellation Autoencoder



The set transformer encoder h^{caps} predicts parameters of two object capsules, which predict affine transformations, precisions and presences of object and part capsules. Finally, input points are explained by a mixture of predictions, where the size of the circle corresponds to its precision.

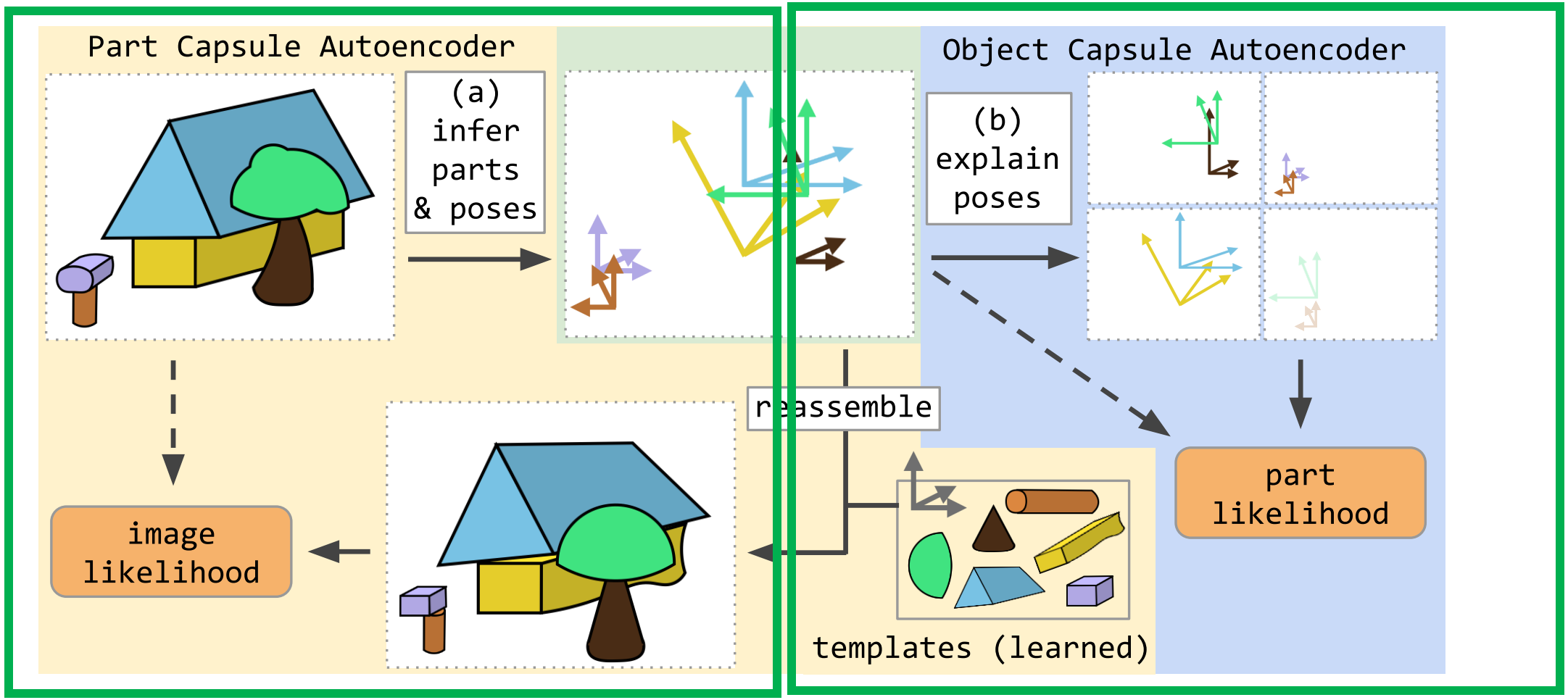
Constellation Autoencoder

- The model is trained to reconstruct the points (top row) under the CCAE mixture model.
- The bottom row colors the points based on the parent with the highest posterior probability in the mixture model.
- The right-most column shows a failure case.



Unsupervised segmentation of points belonging to up to three constellations of squares and triangles at different positions, scales and orientations.

Overview



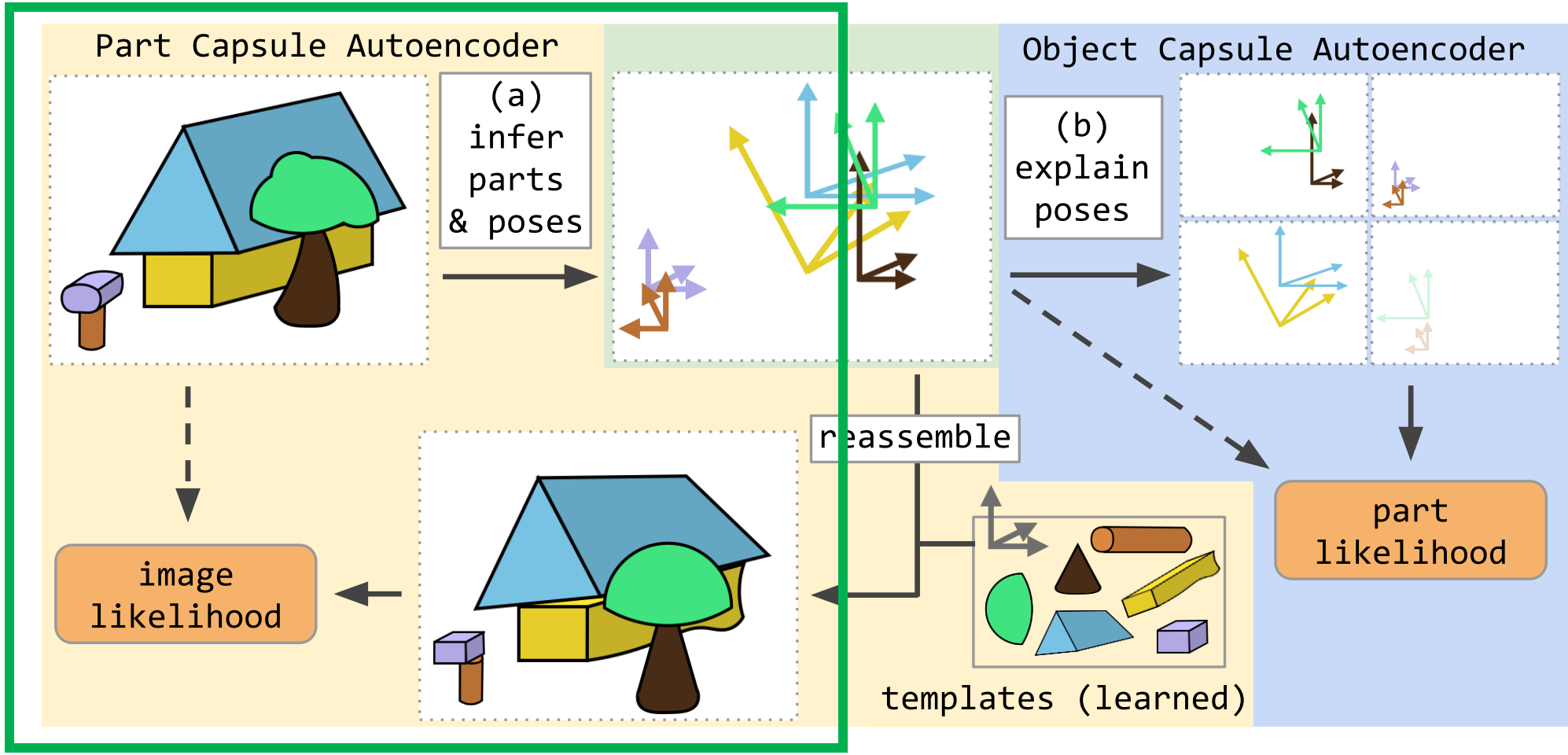
The Stacked Capsule Autoencoder (SCAE) is composed of a Part Capsule Autoencoder (PCAE) followed by an Object Capsule Autoencoder (OCAE). It can decompose an image into its parts and group parts into objects

Credits: http://akosiorek.github.io/ml/2019/06/23/stacked_capsule_autoencoders.html

Part Capsule AE

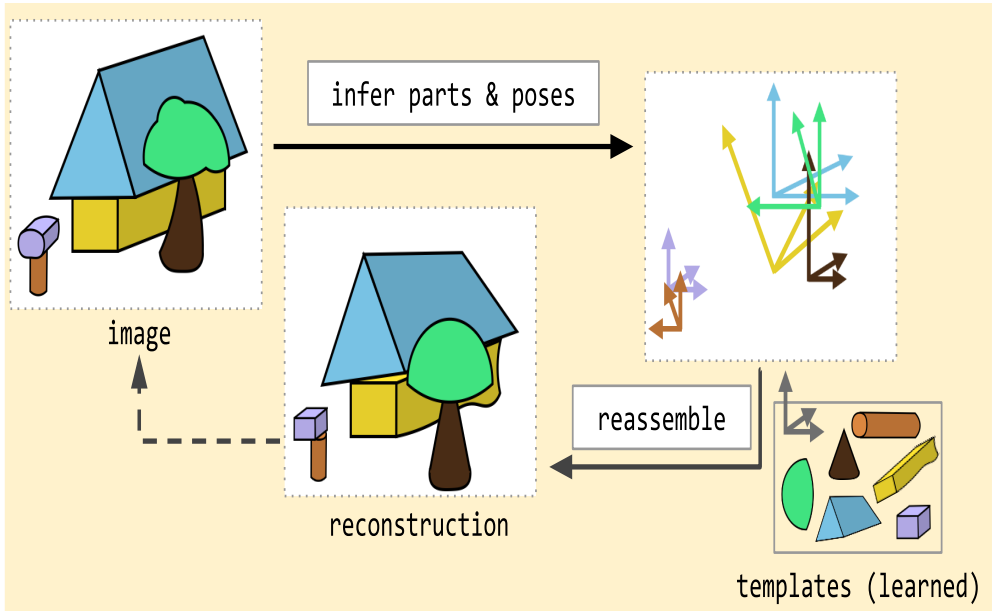


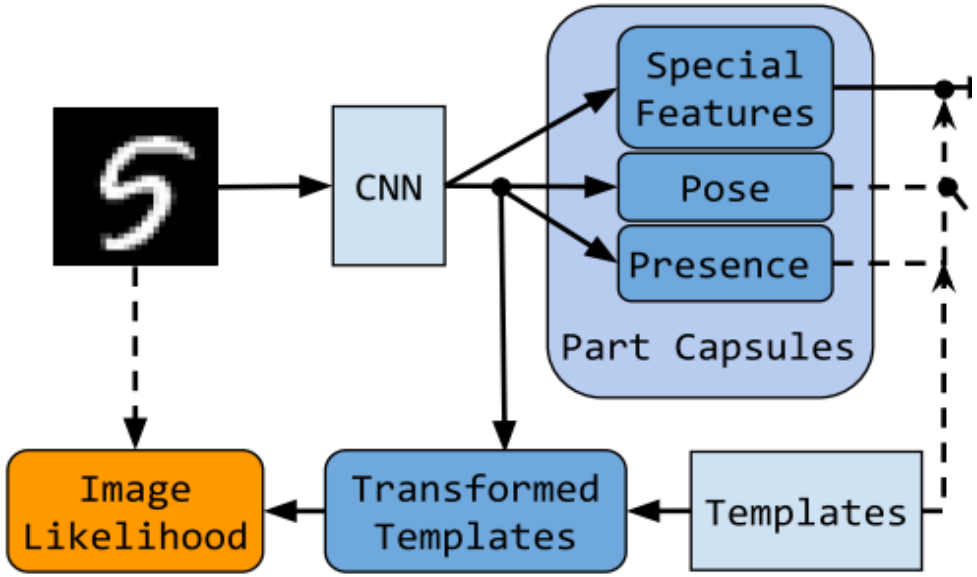
Overview



The Stacked Capsule Autoencoder (SCAE) is composed of a Part Capsule Autoencoder (PCAE) followed by an Object Capsule Autoencoder (OCAE). It can decompose an image into its parts and group parts into objects

Part Capsule Autoencoder (PCAE)

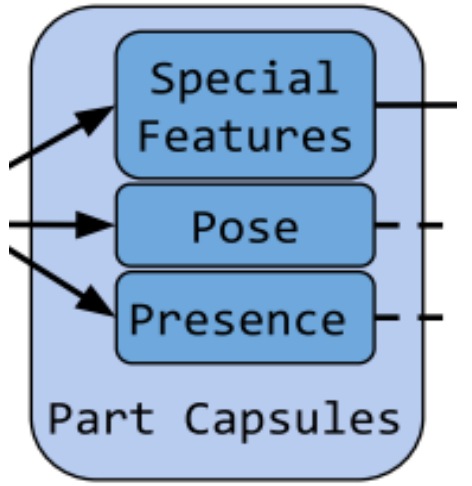




PCAE detects parts and their poses from the image and reconstructs the image by directly assembling it from affine-transformed parts.

Part Capsule Autoencoder (PCAE)

- Explaining images as geometrical arrangements of parts requires
  - Discovering parts
  - Inferring the relationships of the parts
- Each part is a capsule
  - a six-dimensional pose x_m (two rotations, two translations, scale and shear), a presence variable d_m



PCAE

$$\mathbf{x}_{1:M}, d_{1:M}, \mathbf{z}_{1:M} = \mathrm{h}^{\mathrm{enc}}(\mathbf{y})$$ predict part capsule parameters,

$$\boldsymbol{c}_m = \mathrm{MLP}(\mathbf{z}_m)$$ predict the color of the mth template,

$$\widehat{T}_m = \mathrm{TransformImage}(T_m, \mathbf{x}_m)$$ apply affine transforms to image templates

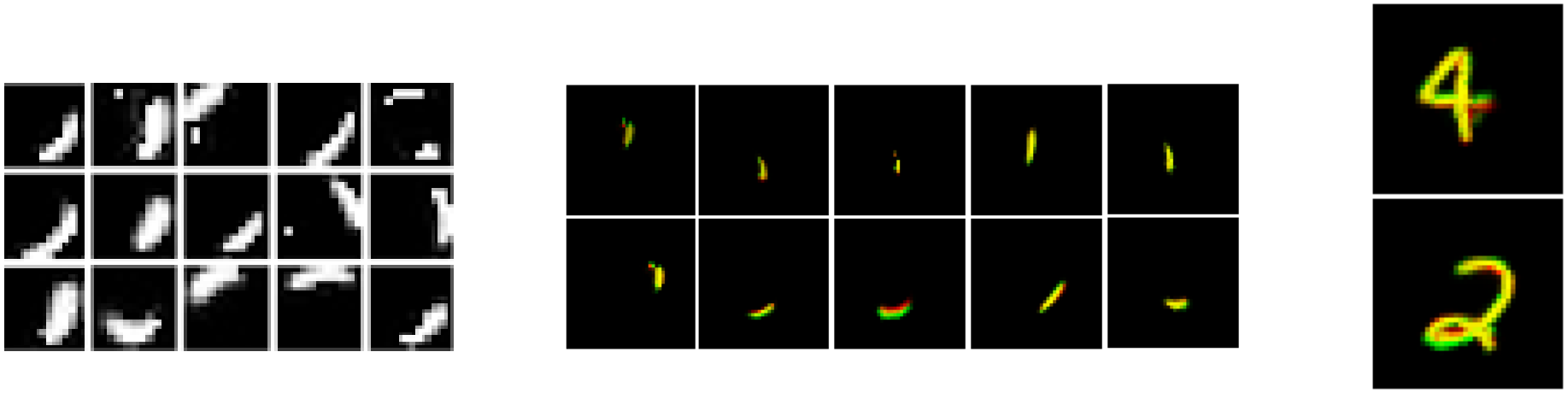
$$p^y_{m,i,j} \propto d_m \widehat{T}^a_{m,i,j}$$ compute mixing probabilities,

$$p(\mathbf{y}) = \prod_{i,j} \sum_{m=1}^{M} p^y_{m,i,j} \mathcal{N}\left(y_{i,j} \mid \boldsymbol{c}_m \cdot \widehat{T}^c_{m,i,j}; \sigma_y^2\right)$$ calculate the image likelihood.



PCAE

- Part-discovery problem as auto-encoding: the encoder learns to model parts and their poses

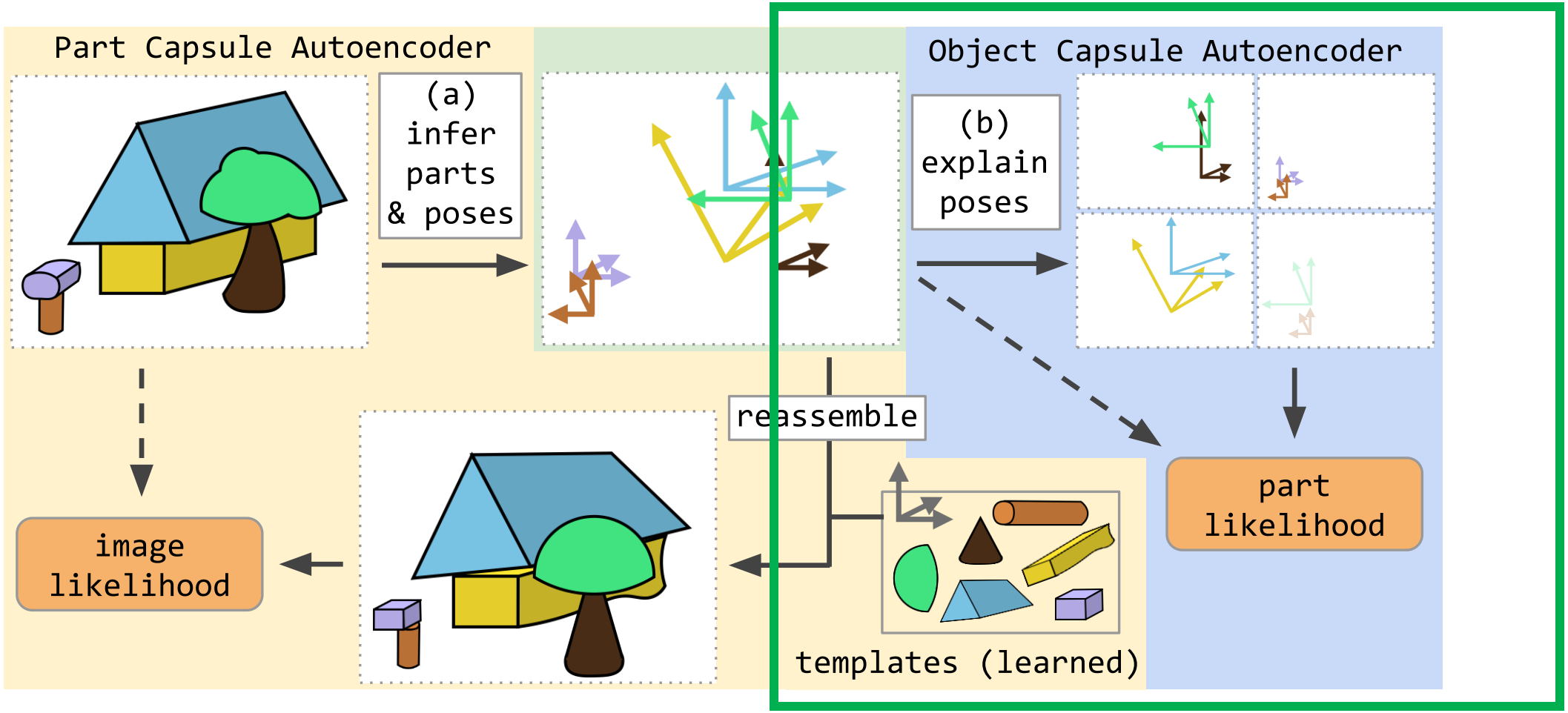


Stroke-like templates learned on MNIST (left) as well as sobel-filtered SVHN (middle) and CIFAR10 (right). For SVHN they often take the form of double strokes due to sobel filtering

Object Capsule AE



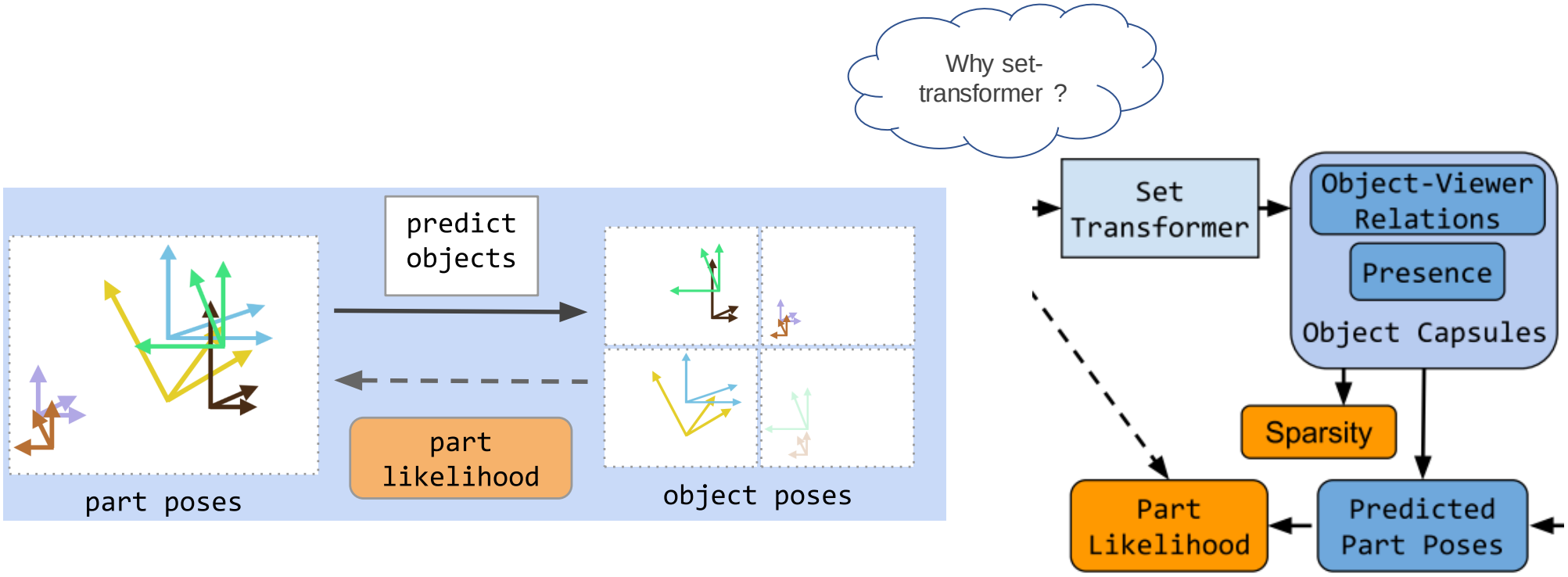
Overview



The Stacked Capsule Autoencoder (SCAE) is composed of a Part Capsule Autoencoder (PCAE) followed by an Object Capsule Autoencoder (OCAE). It can decompose an image into its parts and group parts into objects

Credits: http://akosiorek.github.io/ml/2019/06/23/stacked_capsule_autoencoders.html

Object Capsule Autoencoder (OCAE)

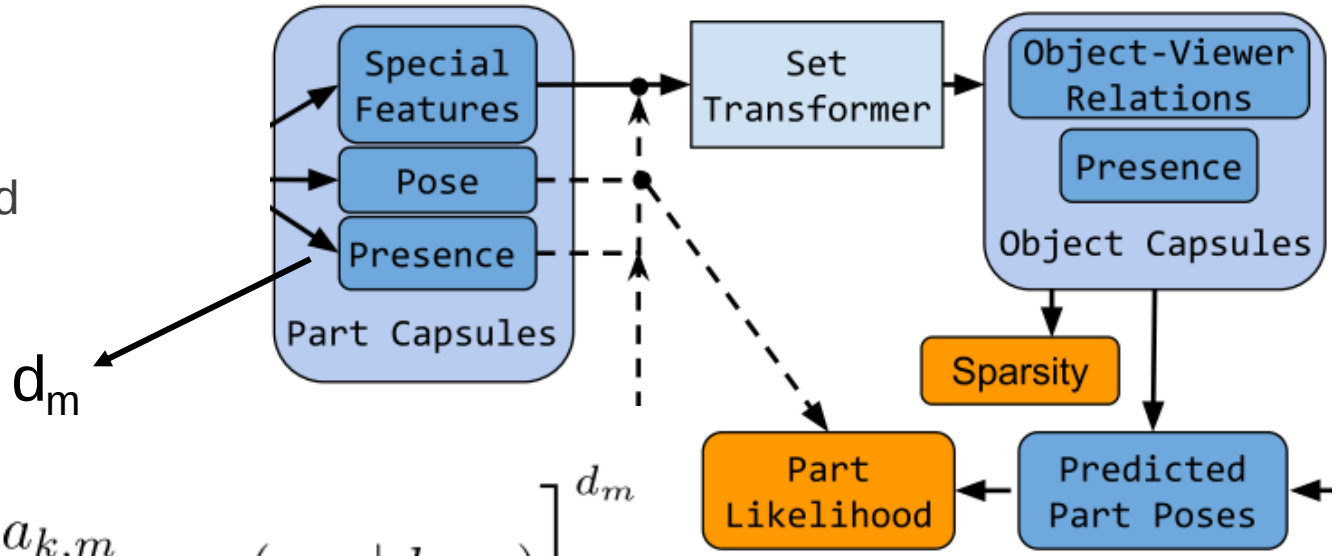


OCAE tries to explain part poses as a sparse set of objects, where every present object predicts several parts. It automatically discovers structure in the data, whereby different object capsules specialize to different objects.

Object Capsule Autoencoder

- Input - Poses x_m, special features z_m and flattened templates T_m (which convey the identity of the part capsule)

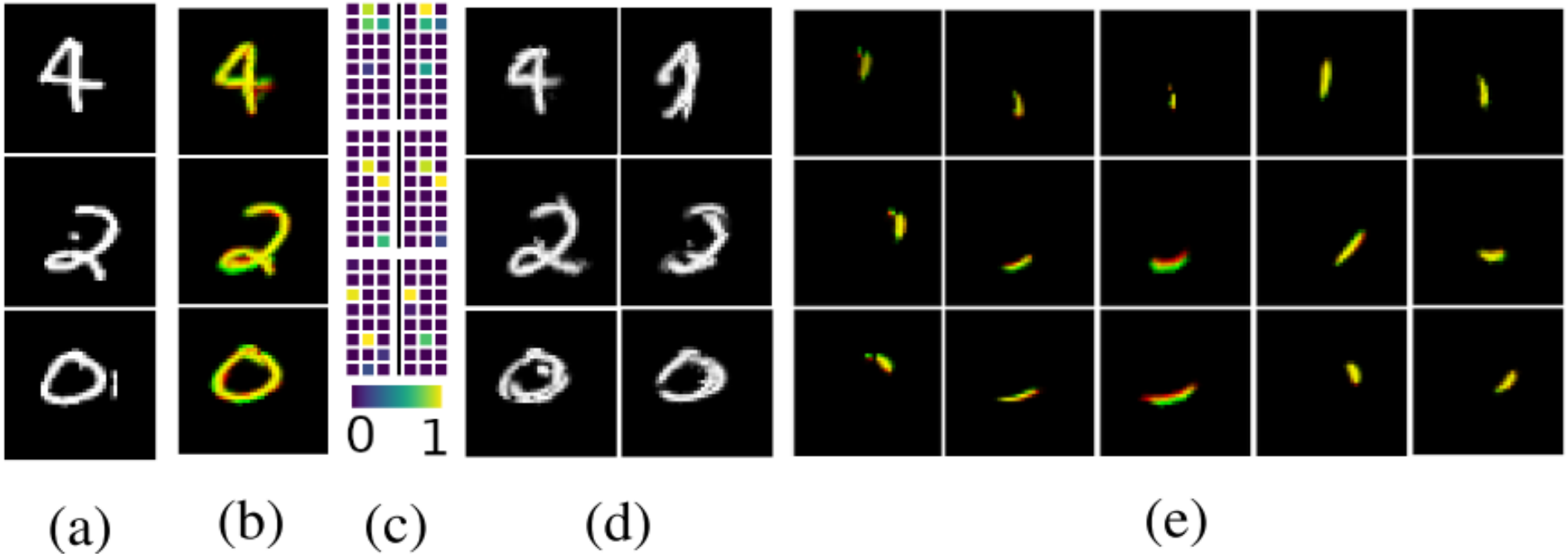
  - Part capsule presence prob d_m fed to OCAE's encoder.
  - d_ms are also used to weigh the part-capsules' log-likelihood, so that we do not take log-likelihood of absent points into account.



Part-capsule likelihood is given by

$$p(\mathbf{x}_{1:M}, d_{1:M}) = \prod_{m=1}^{M} \left[\sum_{k=1}^{K} \frac{a_k a_{k,m}}{\sum_i a_i \sum_j a_{i,j}} p(\mathbf{x}_m \mid k, m) \right]^{d_m}$$

Object Capsule Autoencoder



MNIST (a) images, (b) reconstructions from part capsules in red and object capsules in green, with overlapping regions in yellow. Only a few object capsules are activated for every input (c) a priori (left) and even fewer are needed to reconstruct it (right). The most active capsules (d) capture object identity and its appearance; (e) shows a few of the affine-transformed templates used for reconstruction.

Sparse and Diverse Capsules

Prior and posterior object-capsule presence

To specialize object-capsules to particular arrangements of parts.

$$a_k^{\text{prior}} = a_k \max_m a_{m,k}\,, \qquad a_{k,m}^{\text{posterior}} = a_k a_{k,m}\,\mathcal{N}(\mathbf{x}_m \mid m, k)$$

With K Object Capsules and M Part capsules.

maximum presence probability among predictions from object capsule k

unnormalized mixing proportion used to explain part capsule m

Sparse and Diverse Capsules

- Prior

$$\overline{u}_k = \sum_{b=1}^{B} a_{b,k}^{\text{prior}}$$

sum of presence probabilities of the object capsule k among different training samples for minibatch B

$$\widehat{u}_b = \sum_{k=1}^{K} a_{b,k}^{\text{prior}}$$

sum of object capsule presence probabilities for given sample

$$\mathcal{L}_{\text{prior}} = \frac{1}{B}\sum_{b=1}^{B}\left(\widehat{u}_b - {}^{K}/_{C}\right)^2 + \frac{1}{K}\sum_{k=1}^{K}\left(\overline{u}_k - {}^{B}/_{C}\right)^2$$

K –samples K/C capsules
C – Classes B/C - sum of prob for object capsule
B- Minibatch size

- Posterior

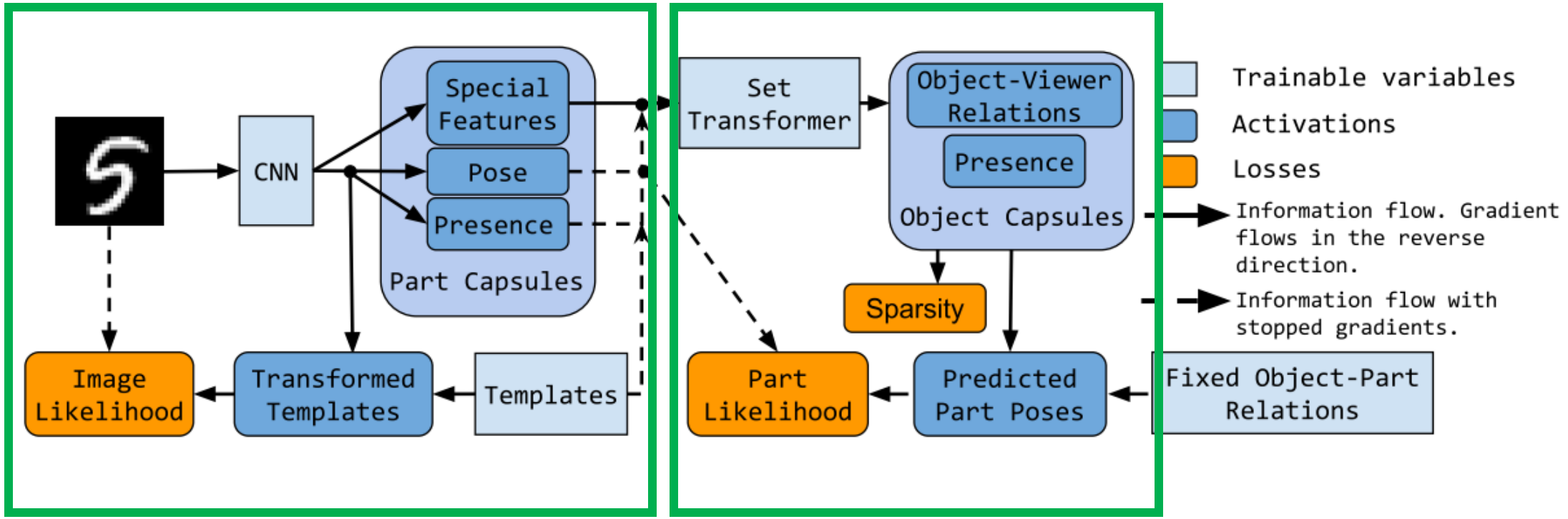
$$\overline{v}_k = \sum_{k,m} a_{b,k,m}^{\text{posterior}}$$

$$\widehat{v}_b = \sum_{b,m} a_{b,k,m}^{\text{posterior}}$$

$$\mathcal{L}_{\text{posterior}} = \frac{1}{K}\sum_{k=1}^{K}\mathcal{H}(\overline{v}_k) - \frac{1}{B}\sum_{b=1}^{B}\mathcal{H}(\widehat{v}_b)$$

Ablation study shows that the model can perform equally well without these posterior sparsity constraints

Overview

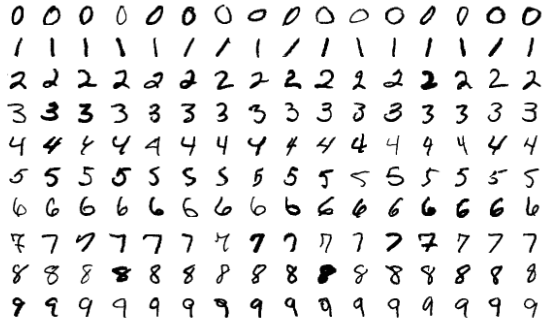

CNN
Special Features
Pose
Presence
Part Capsules
Image Likelihood
Transformed Templates
Templates
Set Transformer
Object-Viewer Relations
Presence
Object Capsules
Sparsity
Part Likelihood
Predicted Part Poses
Fixed Object-Part Relations
Trainable variables
Activations
Losses
Information flow. Gradient flows in the reverse direction.
Information flow with stopped gradients.

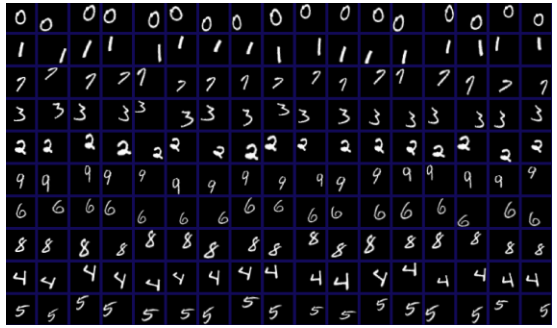
Datasets and Evaluation



Datasets

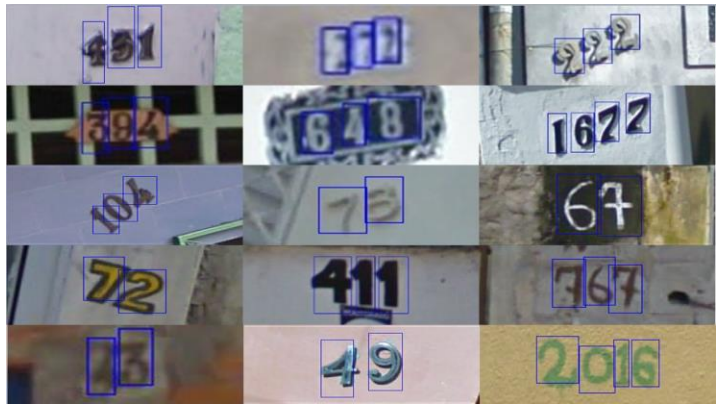
- MNIST
  - Handwritten digit
  - training set – 60k; test set - 10k examples
  - Size - 28 x 28 pixels
- 40×40 MNIST
  - Padded and translated original NIST images
- AFFNIST
  - Applying various reasonable affine transformations to MNIST
  - Train set - 50k; Test set – 10k; validation set - 10k images

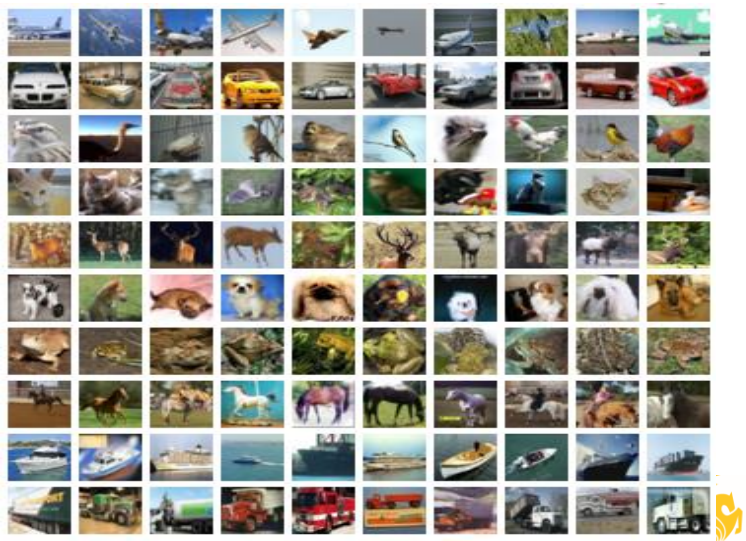




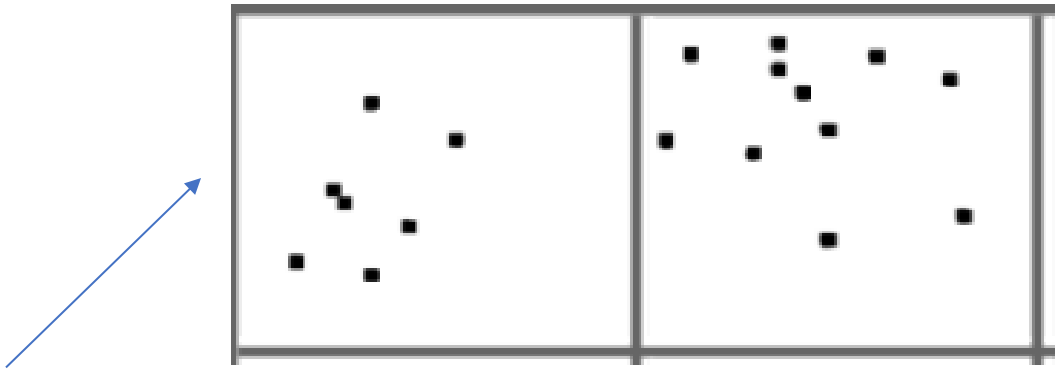
Datasets

- SVHN
  - 10 classes
  - Over 600k Real-World Images of House Numbers From Google Street View Images
  - Size - 32 x 32 pixels
- CIFAR-10
  - Training set – 50k; test set – 10k
  - 32x32 images in 10 classes





Evaluation



- Generated Constellations
  - input example consists of up to 11 2D points belonging to up to three different constellations (two squares and a triangle)
  - binary variables indicating the presence of the points

- CCAE compared against a baseline that uses the same encoder but a simpler decoder

Method	% error
Baseline (simple decoder)	26
CCAE	2.8

Experiments

Method	MNIST	CIFAR10	SVHN
KMEANS (Haeusser et al., 2018)	53.49	20.8	12.5
AE (Bengio et al., 2007)§	81.2	31.4	-
GAN (Radford et al., 2016)§	82.8	31.5	-
IMSAT (Hu et al., 2017)†,∇	**98.4** (0.4)	45.6 (0.8)	**57.3** (3.9)
IIC (Ji et al., 2018)§,†	**98.4** (0.6)	**57.6** (5.0)	-
ADC (Haeusser et al., 2018)†	**98.7** (0.6)	29.3 (1.5)	38.6 (4.1)
SCAE (LIN-MATCH)	**98.7 (0.35)**	25.01 (1.0)	**55.33 (3.4)**
SCAE (LIN-PRED)	**99.0 (0.07)**	33.48 (0.3)	**67.27 (4.5)**

Unsupervised classification results in % with (standard deviation) are averaged over 5 runs.
Results marked with † use data augmentation, ∇ use IMAGENET-pretrained features instead of images, while § are taken from Ji et al., 2018[3].

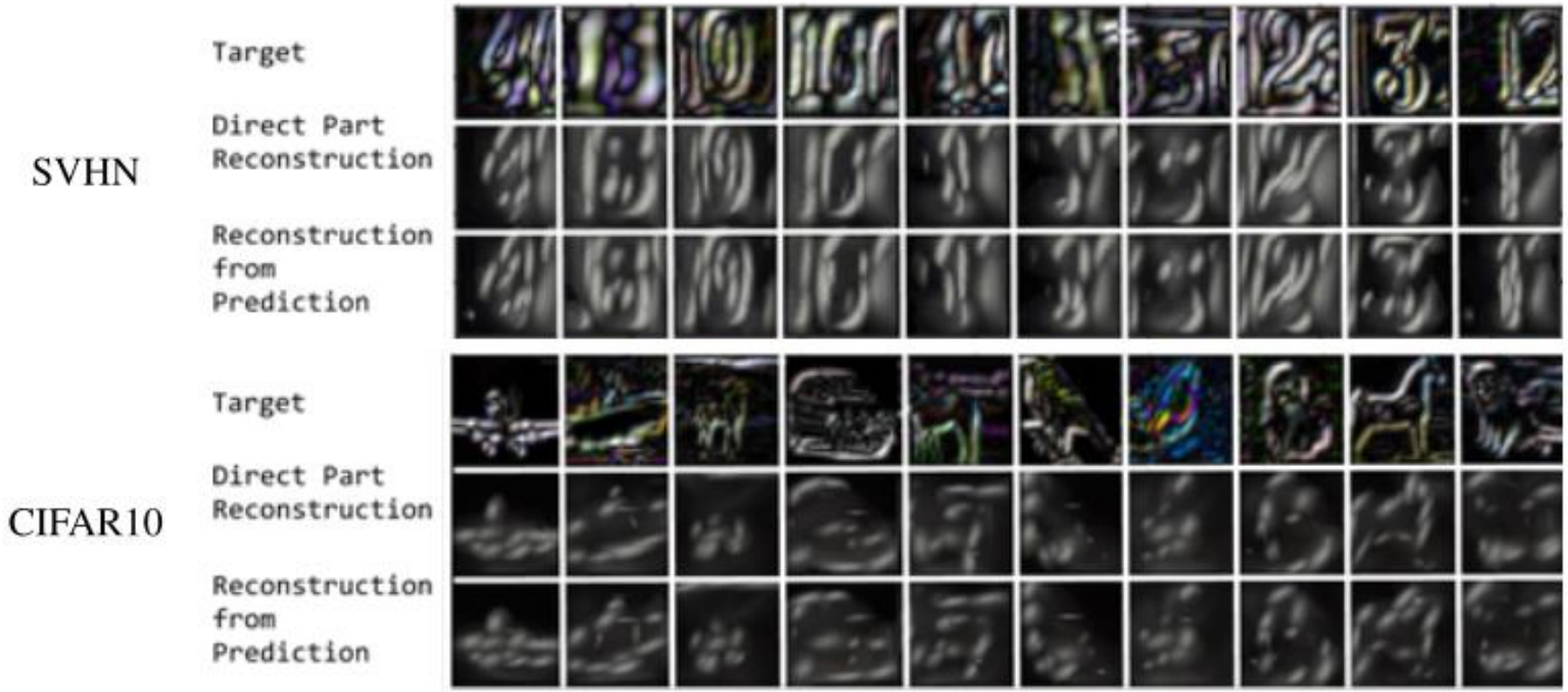
Experiments

	Method	MNIST	40 × 40 MNIST	AFFNIST
	full model	**95.3 (4.65)**	**98.7 (0.35)**	**92.2 (0.59)**
a)	no posterior sparsity	**97.5 (1.55)**	**95.0 (7.20)**	**85.3 (11.67)**
	no prior sparsity	72.4 (22.39)	88.2 (6.98)	71.3 (5.46)
	no prior/posterior sparsity	84.7 (3.01)	82.0 (5.46)	59.0 (5.66)
b)	no noise in object caps	**96.7 (2.30)**	**98.5 (0.12)**	**93.5 (0.38)**
	no noise in any caps	**93.1 (5.09)**	78.5 (22.69)	64.1 (26.74)
	no noise in part caps	**93.9 (7.16)**	82.8 (24.83)	70.7 (25.96)
c)	similarity transforms	**97.5 (1.55)**	95.9 (1.59)	88.9 (1.58)
	no deformations	87.3 (21.48)	87.2 (18.54)	79.0 (22.44)
d)	LINEAR part enc	**98.0 (0.52)**	63.2 (31.47)	50.8 (26.46)
	CONV part enc	**97.6 (1.22)**	**97.8 (.98)**	81.6 (1.66)
e)	MLP enc for object caps	27.1 (9.03)	36.3 (3.70)	25.29 (3.69)
f)	no special features	90.7 (2.25)	58.7 (31.60)	44.5 (21.71)

Ablation study on MNIST. AFFNIST results show out-of-distribution generalization properties and come from a model trained on 40×40 MNIST. Numbers represent average % and (standard deviation) over 10 runs.



Experiments



10 Sample SVHN and Cifar10 reconstructions. First row shows Sobel filtered target image. Second row shows the reconstruction from Part Capsule Layer directly. Third row shows the reconstruction if we use the object predictions for the Part poses instead of Part poses themselves for reconstruction. The SCAE model is trained completely unsupervised but the reconstructions tend to focus on the center digit in SVHN and filter the rest of the clutter

Conclusion

SCAE defines a new method for representation learning, where an arbitrary encoder learns viewpoint-equivariant representations by inferring parts and their poses and groups them into objects.



References

1. A. Kosiorek, H. Kim, Y. W. Teh, and I. Posner. “Sequential Attend, Infer, Repeat: Generative modelling of moving objects”. In: NeurIPS 2018
2. C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. “MONet: Unsupervised Scene Decomposition and Representation”. In: CoRR 2019.
3. X. Ji, J. F. Henriques, and A. Vedaldi. “Invariant Information Distillation for Unsupervised Image Segmentation and Clustering”. In: CoRR 2018
4. R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, A. Trischler, and Y. Bengio. “Learning Deep Representations by Mutual Information Estimation and Maximization”. In: CoRR 2019.
5. G. E. Hinton, A. Krizhevsky, and S. D. Wang. “Transforming Auto-Encoders”. In: International Conference on Artifical Neural Networks 2011.

Questions ?



Thank you!

