

Master's Thesis in Robotics, Cognition, Intelligence

Evaluation and Generalization of Capsule Networks in Neurorobotics

Jean Elsner

Submission Date 29 November 2018

Table of Contents

1. Motivation of Capsule Network Architecture
2. Formalizing Generalization
3. Dataset Generation
4. Results



Limitations of Deep Learning

Convolutional neural networks (CNNs) are the current state-of-the-art deep learning approach in computer vision. They do however suffer several **drawbacks** that limit their use especially in **real-time** applications (robotics).

- Invariant only to variations in position, not affine transformations in general
- Training artificial neural networks is **computationally very expensive** and can take several days or even weeks
- **High latency** during inference, **high energy** requirements limit mobile/real-time applications
- Creation of **large labelled datasets** necessary, covering all variations in lighting, color, viewpoints etc.

Limitations of Deep Learning

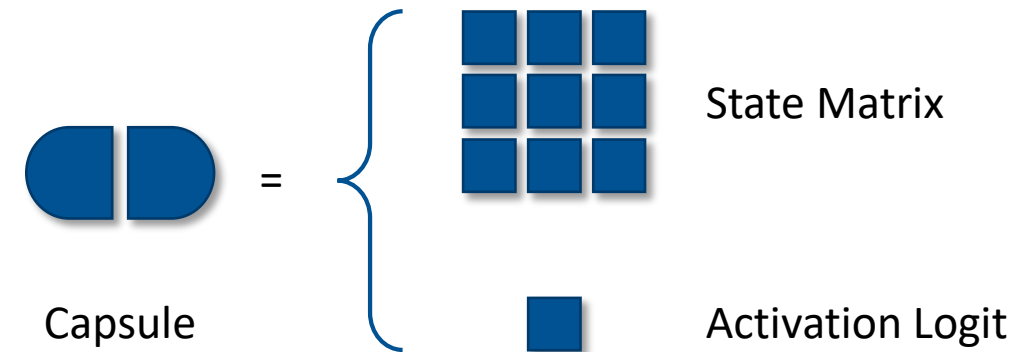
Convolutional neural networks (CNNs) are the current state-of-the-art deep learning approach in computer vision. They do however suffer several **drawbacks** that limit their use especially in **real-time** applications (robotics).

- Invariant only to variations in position, not affine transformations in general 
- Training artificial neural networks is **computationally very expensive**, taking days or even weeks
- **High latency** during inference, **high energy** requirements limit their use in **real-time** applications
- Creation of **large labelled datasets** necessary, covering all variations in lighting, color, viewpoints etc. 

Capsule networks are proposed to tackle two of these issues by learning a specific representation!

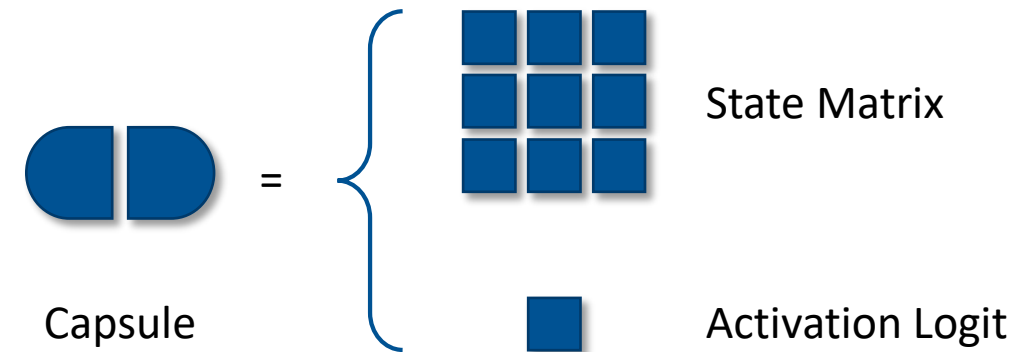
Capsule

A capsule is conceived as a **group of neurons** – a state vector (or matrix) and an activation. Every capsule represents a feature whose **instantiation parameters** and probability of being present are encoded in the state vector and activation respectively.



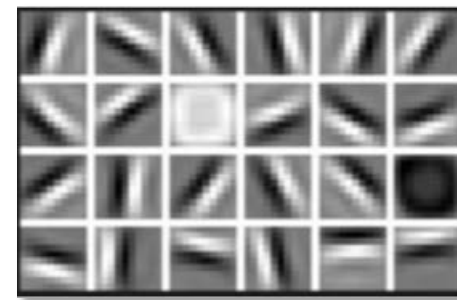
Capsule

A capsule is conceived as a **group of neurons** – a state vector (or matrix) and an activation. Every capsule represents a feature whose **instantiation parameters** and probability of being present are encoded in the state vector and activation respectively.



Instantiation Parameters

The coordinates of a specific instance of an object on the **appearance manifold**. For a simple object like an edge, this could be orientation, strength, etc. For more complex objects these can be pose, lighting, occlusion and many more.



First layer kernels as learned by a CNN

Routing between Capsules

Assuming a layer of capsules already extracted their activation and state from pixel intensities, a **routing algorithm** can be designed to propagate the signal in a capsule network based on **agreement**.

1. Capsules in the lower layer cast votes about what they expect the state of the capsules in the higher level to be, based on their own state
2. The more capsules agree about a higher capsule's state the higher its activation
3. States of higher capsules are computed based on the average votes they received

Routing between Capsules

Assuming a layer of capsules already extracted their activation and state from pixel intensities, a **routing algorithm** can be designed to propagate the signal in a capsule network based on **agreement**.

1. Capsules in the lower layer cast votes about what they expect the state of the capsules in the higher level to be, based on their own state
2. The more capsules agree about a higher capsule's state the higher its activation
3. States of higher capsules are computed based on the average votes they received

Benefits

- No pooling operations → information on position of activation is **not discarded**
- No unprincipled activation functions, rather a **sensible objective function** is minimized
- The network only learns linear transforms W_{ij} between capsule states

Routing between Capsules

For a lower capsule i its **vote** for a higher capsule j is simply calculated as a **matrix multiplication** of its state μ_i and the discriminatively learned matrix transform W_{ij} . The job of the routing algorithm is then to **cluster** votes of the lower layer and set the **expected** activations and states of the higher layer.

EM Routing

The Expectation Maximization algorithm fits a **gaussian mixture model** by iteratively computing a log-likelihood function using the current estimation of the parameters in the **expectation** step and computing parameters **maximizing** the same log-likelihood function in the following maximization step. In EM Routing, the higher capsules are interpreted as **gaussian distributions** while the vectorized pose matrices of the lower capsules constitute the **data points**.

Routing between Capsules

Matrix Capsules with EM Routing

- Minimize the system's free energy by activating capsules based on their description length
- Learns part-whole relationships W_{ij} in feature hierarchies
- Matrix Capsules are equivariant in their instantiation parameters

```

1: procedure EM-ROUTING( $\mathbf{a}, \mathbf{V}, r, l$ )
2:   for all  $i \in \Omega_l$  do
3:     for all  $j \in \Omega_{l+1}$  do
4:        $R_{ij} \leftarrow \frac{1}{|\Omega_{l+1}|}$ 
5:   for  $r$  iterations do
6:     for all  $j \in \Omega_{l+1}$  do
7:       M-STEP( $\mathbf{a}, \mathbf{R}, \mathbf{V}, j$ )
8:     for all  $i \in \Omega_l$  do
9:       E-STEP( $\mu, \sigma, \mathbf{a}, \mathbf{V}, i$ )
10:  return  $\mathbf{a}, \mu$ 
11: procedure M-STEP( $\mathbf{a}, \mathbf{R}, \mathbf{V}, j$ )
12:   for all  $i \in \Omega_l$  do
13:      $R_{ij} \leftarrow R_{ij} a_i$ 
14:   for  $h \leftarrow 1$  to  $H$  do
15:      $\mu_j^h \leftarrow \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}$ 
16:      $(\sigma_j^h)^2 \leftarrow \frac{\sum_i R_{ij} (V_{ij}^h - \mu_j^h)^2}{\sum_i R_{ij}}$ 
17:      $cost^h \leftarrow (\beta_u + \log(\sigma_j^h)) \sum_i R_{ij}$ 
18:    $a_j \leftarrow \text{logistic}(\lambda(\beta_a - \sum_h cost^h))$ 
19:   return  $\mu_j, \sigma_j$ 
20: procedure E-STEP( $\mu, \sigma, \mathbf{a}, \mathbf{V}, i$ )
21:   for all  $j \in \Omega_{l+1}$  do
22:      $p_j \leftarrow \frac{1}{\prod_h 2\pi(\sigma_j^h)^2} \exp\left(-\sum_h \frac{(V_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2}\right)$ 
23:      $R_{ij} \leftarrow \frac{\exp(a_j p_j)}{\sum_{k \in \Omega_{l+1}} \exp(a_k p_k)}$ 
24:   return  $\mathbf{R}$ 

```

Routing between Capsules

Matrix Capsules with EM Routing

- Minimize the system's free energy by activating capsules based on their description length
- Learns part-whole relationships W_{ij} in feature hierarchies
- Matrix Capsules are equivariant in their instantiation parameters

Cost for description length →

```

1: procedure EM-ROUTING( $\mathbf{a}, \mathbf{V}, r, l$ )
2:   for all  $i \in \Omega_l$  do
3:     for all  $j \in \Omega_{l+1}$  do
4:        $R_{ij} \leftarrow \frac{1}{|\Omega_{l+1}|}$ 
5:   for  $r$  iterations do
6:     for all  $j \in \Omega_{l+1}$  do
7:       M-STEP( $\mathbf{a}, \mathbf{R}, \mathbf{V}, j$ )
8:     for all  $i \in \Omega_l$  do
9:       E-STEP( $\mu, \sigma, \mathbf{a}, \mathbf{V}, i$ )
10:  return  $\mathbf{a}, \mu$ 
11: procedure M-STEP( $\mathbf{a}, \mathbf{R}, \mathbf{V}, j$ )
12:  for all  $i \in \Omega_l$  do
13:     $R_{ij} \leftarrow R_{ij} a_i$ 
14:  for  $h \leftarrow 1$  to  $H$  do
15:     $\mu_j^h \leftarrow \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}$ 
16:     $(\sigma_j^h)^2 \leftarrow \frac{\sum_i R_{ij} (V_{ij}^h - \mu_j^h)^2}{\sum_i R_{ij}}$ 
17:     $cost^h \leftarrow (\beta_u + \log(\sigma_j^h)) \sum_i R_{ij}$ 
18:   $a_j \leftarrow \text{logistic}(\lambda(\beta_a - \sum_h cost^h))$ 
19:  return  $\mu_j, \sigma_j$ 
20: procedure E-STEP( $\mu, \sigma, \mathbf{a}, \mathbf{V}, i$ )
21:  for all  $j \in \Omega_{l+1}$  do
22:     $p_j \leftarrow \frac{1}{\prod_h 2\pi(\sigma_j^h)^2} \exp\left(-\sum_h \frac{(V_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2}\right)$ 
23:     $R_{ij} \leftarrow \frac{\exp(a_j p_j)}{\sum_{k \in \Omega_{l+1}} \exp(a_k p_k)}$ 
24:  return  $\mathbf{R}$ 

```

Routing between Capsules

Matrix Capsules with EM Routing

- Minimize the system's free energy by activating capsules based on their description length
- Learns part-whole relationships W_{ij} in feature hierarchies
- Matrix Capsules are equivariant in their instantiation parameters

Cost for description length →

Probability of seeing the Vote V_{ij} under capsule j →

```

1: procedure EM-ROUTING( $\mathbf{a}, \mathbf{V}, r, l$ )
2:   for all  $i \in \Omega_l$  do
3:     for all  $j \in \Omega_{l+1}$  do
4:        $R_{ij} \leftarrow \frac{1}{|\Omega_{l+1}|}$ 
5:   for  $r$  iterations do
6:     for all  $j \in \Omega_{l+1}$  do
7:       M-STEP( $\mathbf{a}, \mathbf{R}, \mathbf{V}, j$ )
8:     for all  $i \in \Omega_l$  do
9:       E-STEP( $\mu, \sigma, \mathbf{a}, \mathbf{V}, i$ )
10:  return  $\mathbf{a}, \mu$ 
11: procedure M-STEP( $\mathbf{a}, \mathbf{R}, \mathbf{V}, j$ )
12:   for all  $i \in \Omega_l$  do
13:      $R_{ij} \leftarrow R_{ij} a_i$ 
14:   for  $h \leftarrow 1$  to  $H$  do
15:      $\mu_j^h \leftarrow \frac{\sum_i R_{ij} V_{ij}^h}{\sum_i R_{ij}}$ 
16:      $(\sigma_j^h)^2 \leftarrow \frac{\sum_i R_{ij} (V_{ij}^h - \mu_j^h)^2}{\sum_i R_{ij}}$ 
17:      $cost^h \leftarrow (\beta_u + \log(\sigma_j^h)) \sum_i R_{ij}$ 
18:    $a_j \leftarrow \text{logistic}(\lambda(\beta_a - \sum_h cost^h))$ 
19:   return  $\mu_j, \sigma_j$ 
20: procedure E-STEP( $\mu, \sigma, \mathbf{a}, \mathbf{V}, i$ )
21:   for all  $j \in \Omega_{l+1}$  do
22:      $p_j \leftarrow \frac{1}{\prod_h 2\pi(\sigma_j^h)^2} \exp\left(-\sum_h \frac{(V_{ij}^h - \mu_j^h)^2}{2(\sigma_j^h)^2}\right)$ 
23:      $R_{ij} \leftarrow \frac{\exp(a_j p_j)}{\sum_{k \in \Omega_{l+1}} \exp(a_k p_k)}$ 
24:   return  $\mathbf{R}$ 

```

Open Questions

- How to initialize the first capsule layer from pixel intensities?
- Does training matrix capsules with EM routing discriminatively result in the capsules learning useful instantiation parameters?
- How do capsule networks perform compared to established methods?

Open Questions

- How to initialize the first capsule layer from pixel intensities?
→ Hinton et al. suggest using conventional convolutional layers
- Does training matrix capsules with EM routing discriminatively result in the capsules learning useful instantiation parameters?
- How do capsule networks perform compared to established methods?

Formalizing Generalization

Capsules are **equivariant** in their instantiation parameters, i.e. a **nonlinear transform** in pixel intensities results in only a linear transformation in the instantiation parameter's activation. The capsule's **activation probability** remains the same.

- A capsule's ability to generalize is encoded in their instantiation parameters
- Capsule networks can also use convolutional capsule layers, as such they inherently generalize to variations in position like CNNs

Formalizing Generalization

Capsules are **equivariant** in their instantiation parameters, i.e. a **nonlinear transform** in pixel intensities results in only a linear transformation in the instantiation parameter's activation. The capsule's **activation probability** remains the same.

- A capsule's ability to generalize is encoded in their instantiation parameters
- Capsule networks can also use convolutional capsule layers, as such they inherently generalize to variations in position like CNNs

Idea

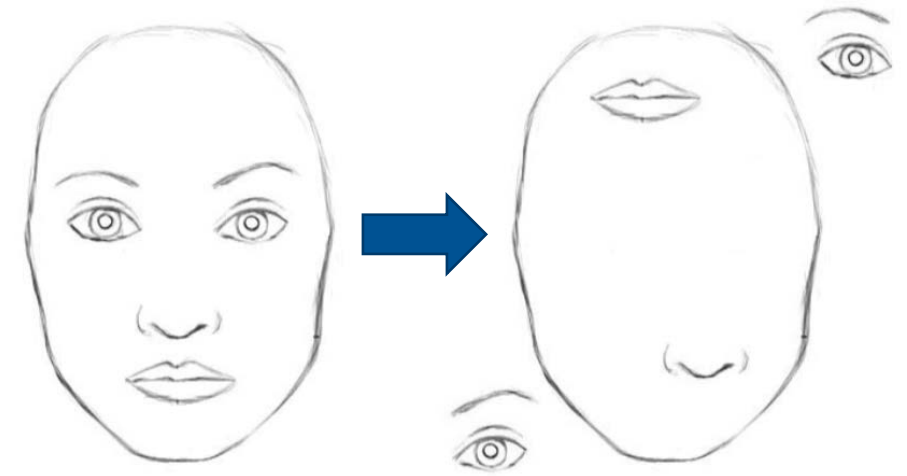
Every instantiation parameter corresponds to a **perturbation** of the data. Parameterize the data to produce perturbed versions across the entire **perturbation space** and evaluate the capsule networks trained on the unperturbed versions.

Example for Perturbation

Capsule networks learn part-whole relationships between features. This means that capsules only activate if the capsules beneath them are in agreement about the state of the instantiation parameters of the higher capsule based on their own instantiation.

Example

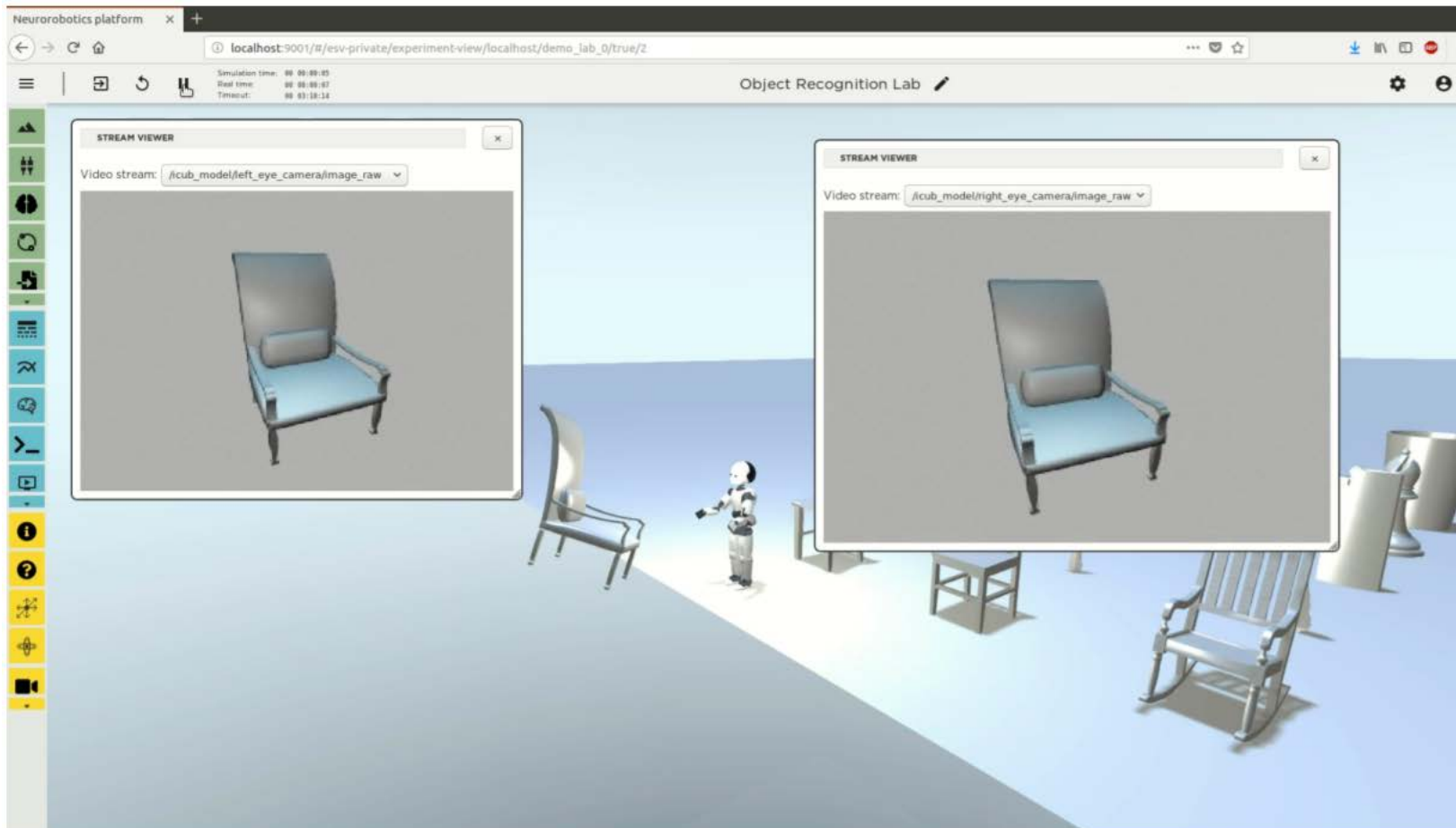
In a capsule network trained to recognize **faces**, capsules in one layer could learn such features as eyes, noses and lips. If the capsules' instantiation represent poses, the part-whole transforms would describe **relative poses**. The part-whole relationships can be deconstructed by placing the features in such a way that the capsules representing them are no longer in agreement. A capsule network, that successfully learned to represent poses, should not respond to the deconstructed version.



Deconstruction of part-whole relationships

Dataset Generation

The [Neurorobotics Platform](#) is used to generate object recognition data.



Dataset Generation

The [Neurorobotics Platform](#) is used to generate object recognition data.

Advantages

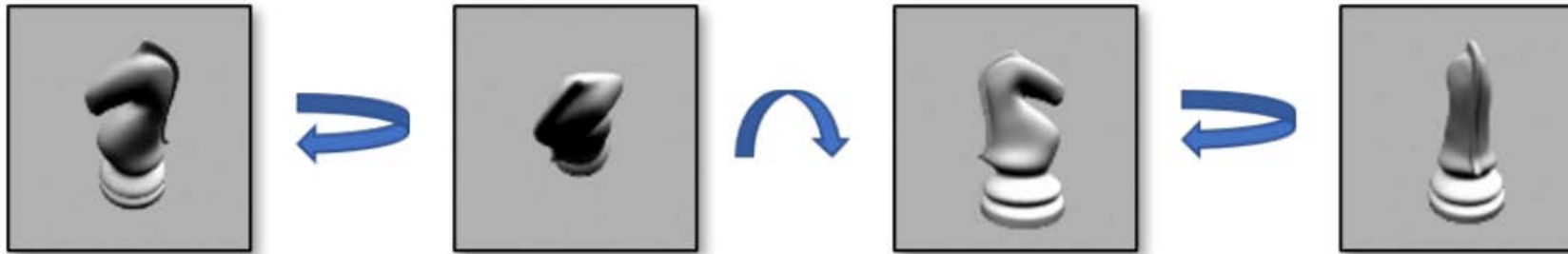
- Large labelled datasets can be create with little “manual labor”
- Supports different modalities by using simulated noisy sensors
- Control perturbations to customize data to test for generalization specifically

Object recognition dataset

Used in thesis: stereoscopic images of 5 everyday object classes

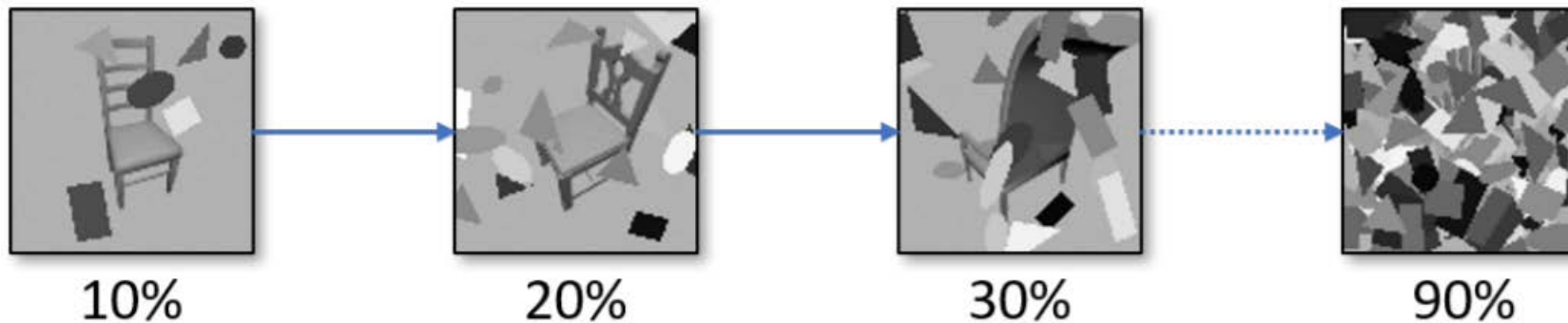


Parameterized Perturbations



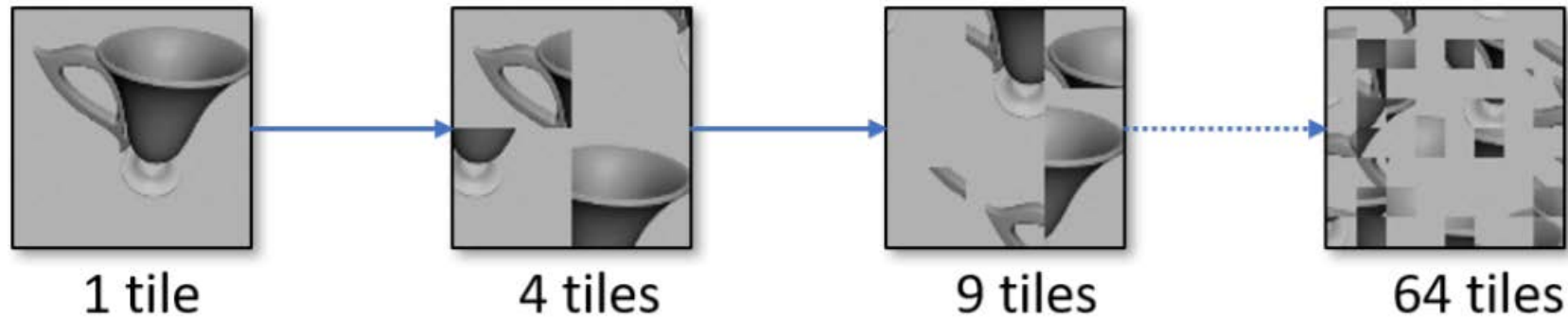
Samples of different viewpoints from the object recognition dataset. The same object from the class of chess figures is viewed from different azimuths and elevations under the same lighting. Note that while the shape of the figure is symmetrical across one axis, the two sides are lit-up differently under different viewpoints

Parameterized Perturbations



Augmentation of the object recognition dataset to test for robustness against occlusion. Ellipses, rectangles and triangles of varying shape and color are randomly placed on the images to occlude the object. An entire dataset is created for each increment in the percentage of coverage.

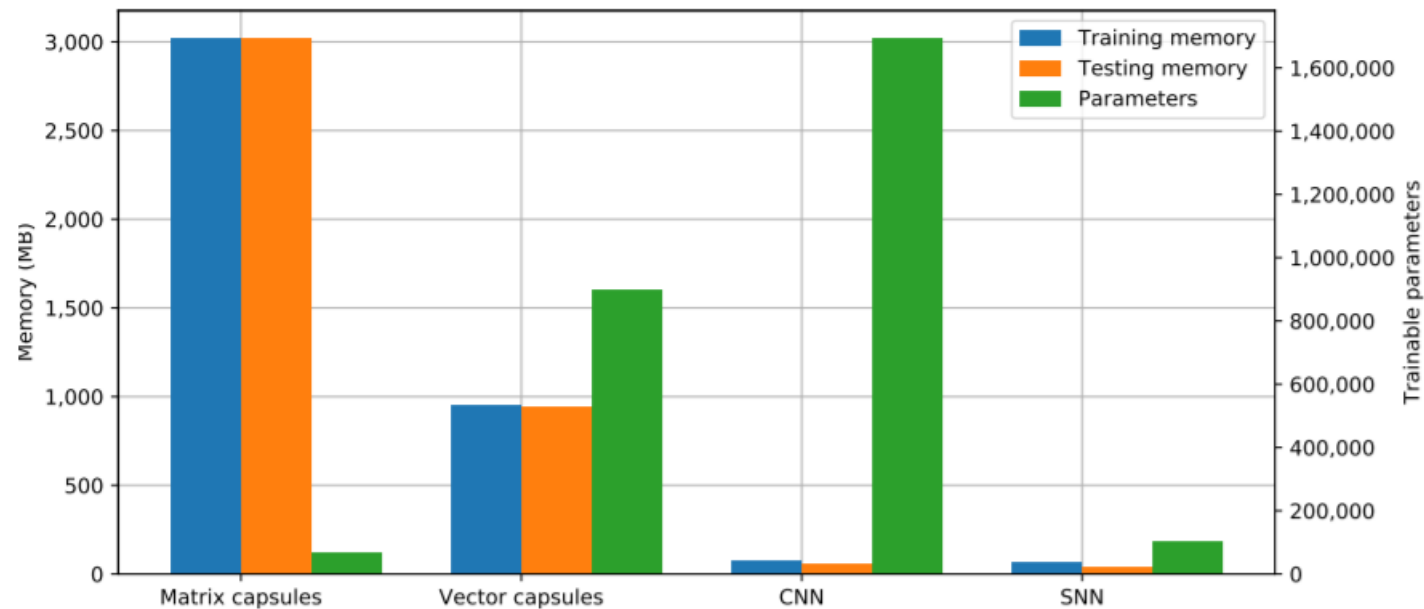
Parameterized Perturbations



Augmentation of the object recognition dataset to test the robustness of part-whole relationships. The images are divided into an increasing number of squares that are randomly permuted. For each number of tiles, an entire dataset is created.

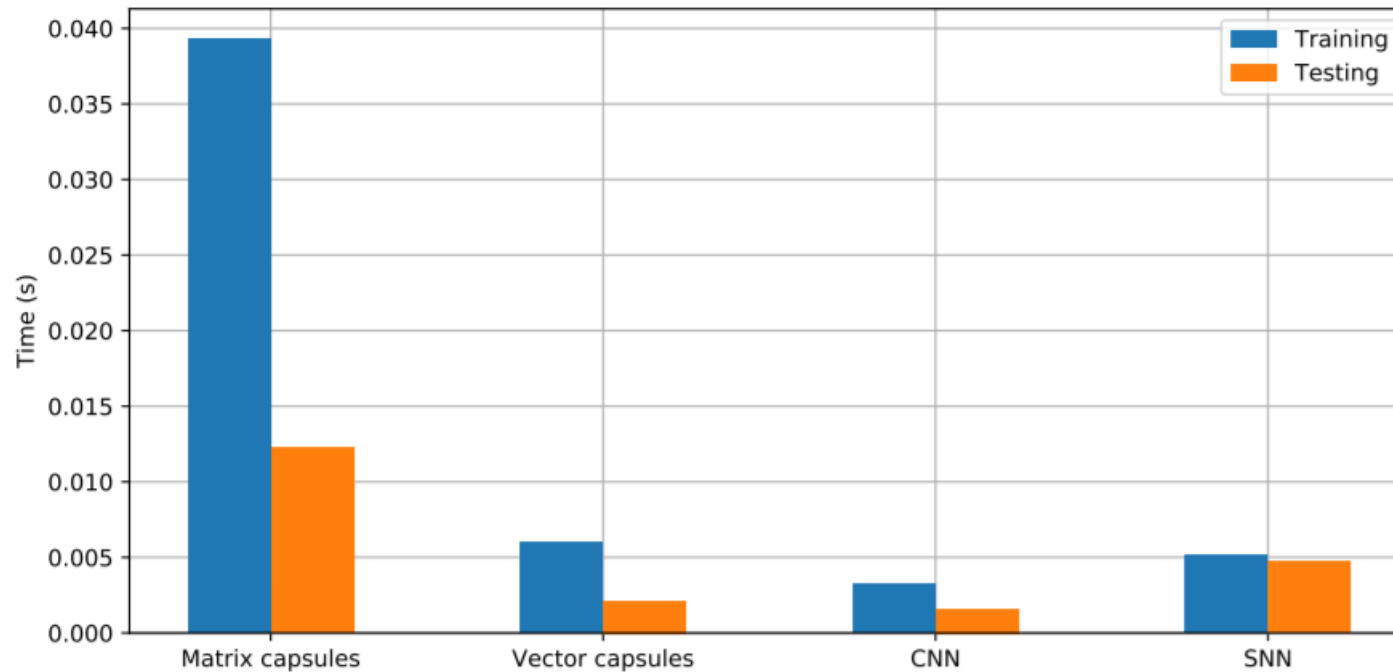
Runtime Results

Matrix Capsules are a **good prior** for how objects are encoded in images, they only need few variables to achieve very good results – but this comes at a **computational cost**.



Runtime Results

Matrix Capsules are a **good prior** for how objects are encoded in images, they only need few variables to achieve very good results – but this comes at a **computational cost**.

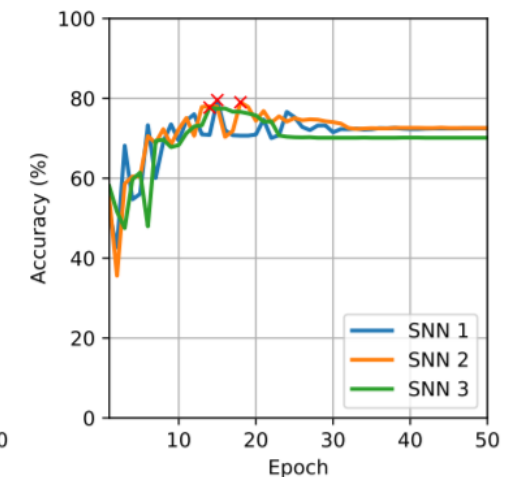
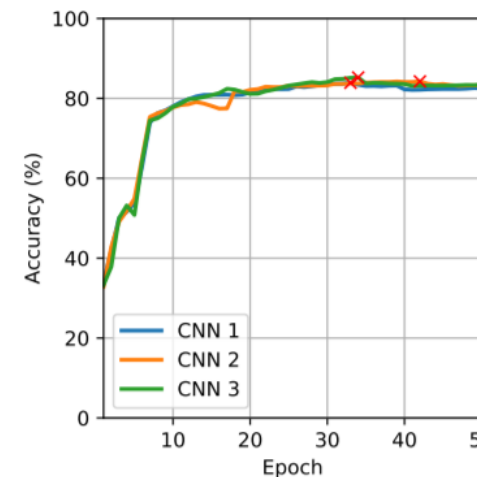
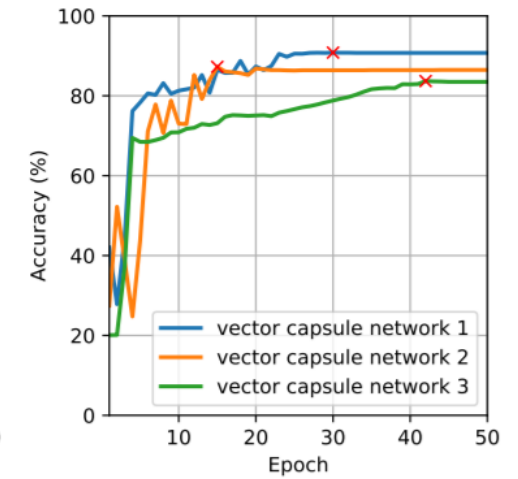
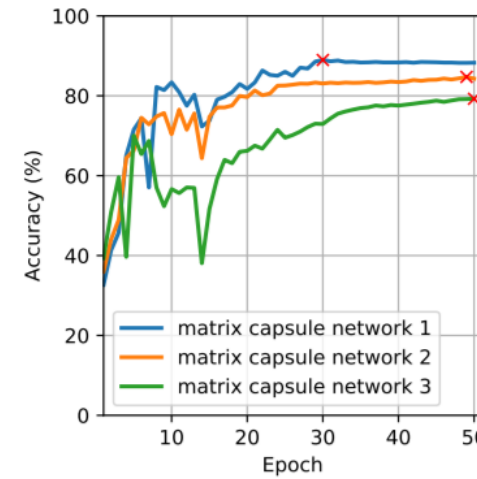


Object Recognition Results

Matrix Capsules are a **good prior** for how objects are encoded in images, they only need few variables to achieve very good results – but this comes at a **computational cost**.

model	batch size	learning rate	routing iterations	accuracy
matrix capsule network	21	9×10^{-3}	2	88.96%
	19	1×10^{-2}	2	84.72%
	21	3×10^{-2}	2	79.25%
vector capsule network	1	2×10^{-4}	2	90.82%
	3	3×10^{-4}	2	87.24%
	2	3×10^{-4}	2	83.69%
CNN	130	1×10^{-5}	-	85.23%
	128	1×10^{-5}	-	84.25%
	131	1×10^{-5}	-	83.90%

hidden neurons	time	learning rate	decay rate	patience	interval	accuracy
100	15	5×10^{-3}	0.7	10	250	79.77%
100	5	5×10^{-3}	0.7	10	250	78.04%
250	15	7×10^{-3}	0.5	10	250	77.72%

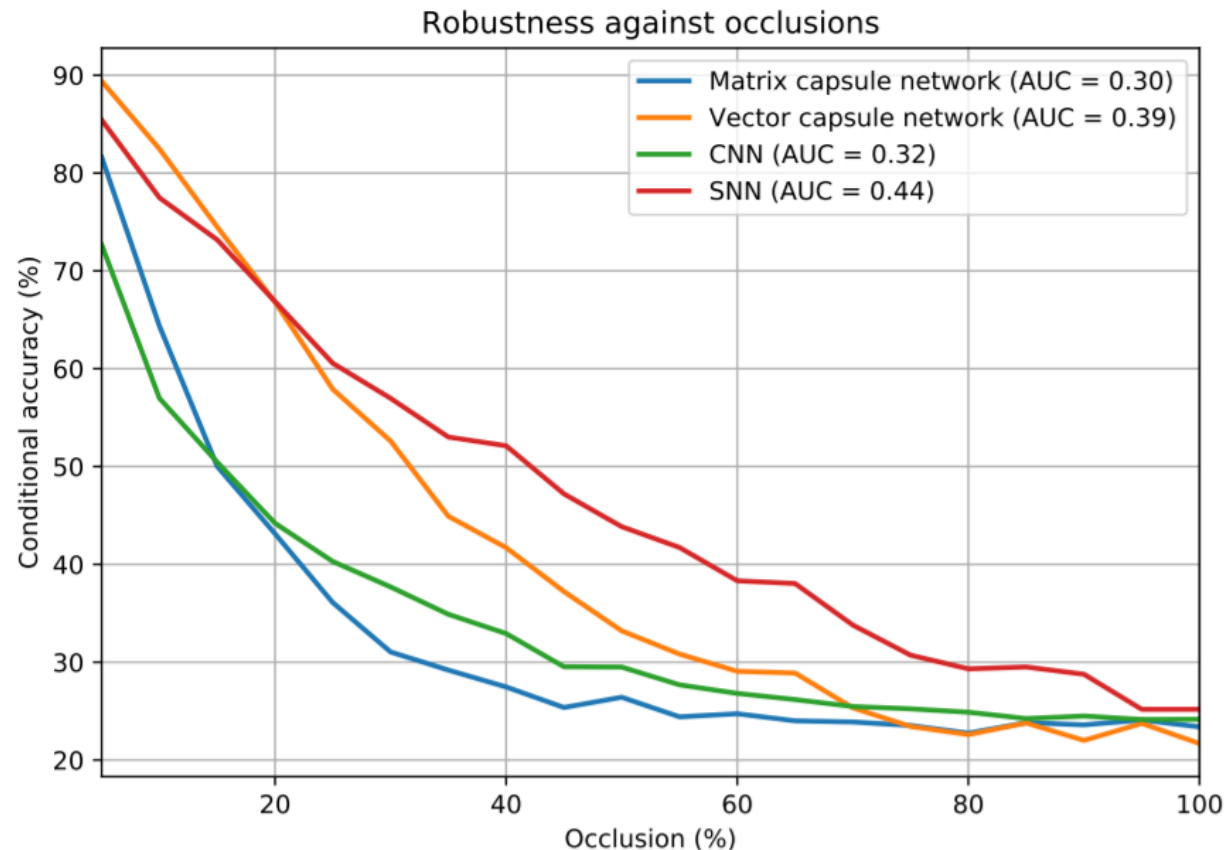


Open Questions

- How to initialize the first capsule layer from pixel intensities?
→ Hinton et al. suggest using conventional convolutional layers
- Does training matrix capsules with EM routing discriminatively result in the capsules learning useful instantiation parameters?
- How do capsule networks perform compared to established methods?
→ Their object recognition accuracy and description length is superior but this achieved using substantially more computational resources

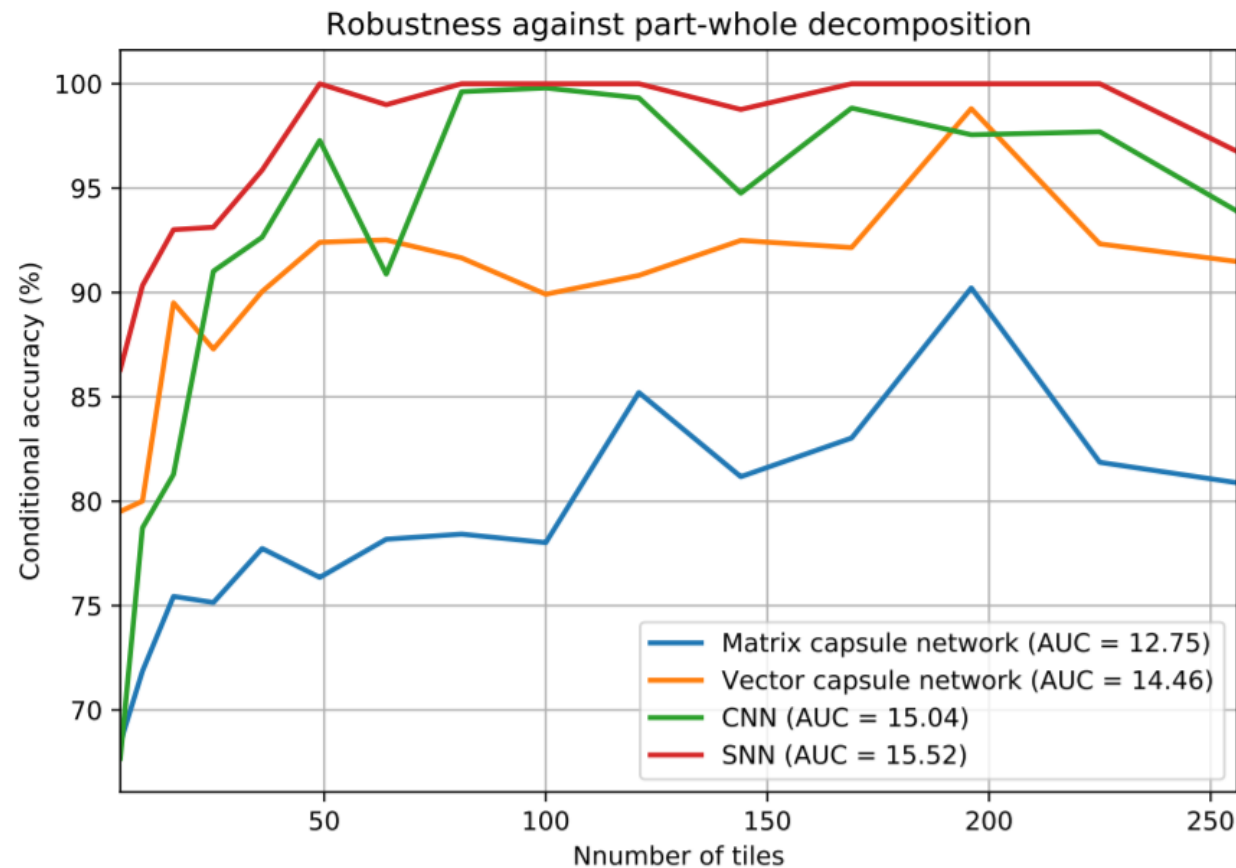
Generalization Results

The capsule networks did not perform particularly well at generalization, suggesting that they did not learn useful instantiation parameters but rather abstract pixel statistics.



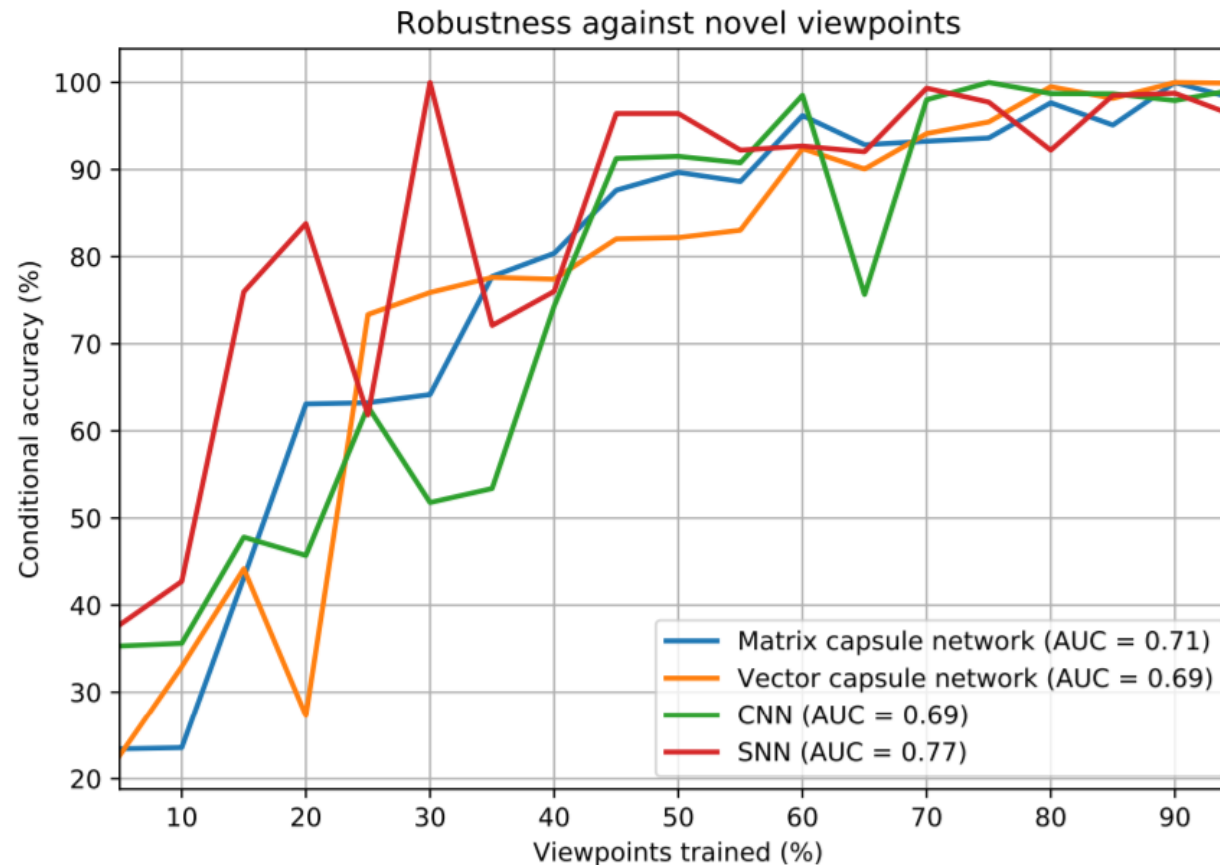
Generalization Results

The capsule networks did not perform particularly well at generalization, suggesting that they did not learn useful instantiation parameters but rather abstract pixel statistics.



Generalization Results

The capsule networks did not perform particularly well at generalization, suggesting that they did not learn useful instantiation parameters but rather abstract pixel statistics.



Conclusion

- How to initialize the first capsule layer from pixel intensities?
→ Hinton et al. suggest using conventional convolutional layers
- Does training matrix capsules with EM routing discriminatively result in the capsules learning useful instantiation parameters?
→ Training capsule networks with stochastic gradient descent using backpropagation does not appear to result in them learning useful representation as they do not generalize
- How do capsule networks perform compared to established methods?
→ Their object recognition accuracy and description length is superior but this achieved using substantially more computational resources

Conclusion

The [thesis](#) as well as the experiment for the [Neurorobotics Platform](#) used to generate the dataset as well as all the [artificial neural network](#) implementations are available at GitHub

<https://github.com/JeanElsner/thesis-capsules-neurorobotics>

Thank you for your attention!