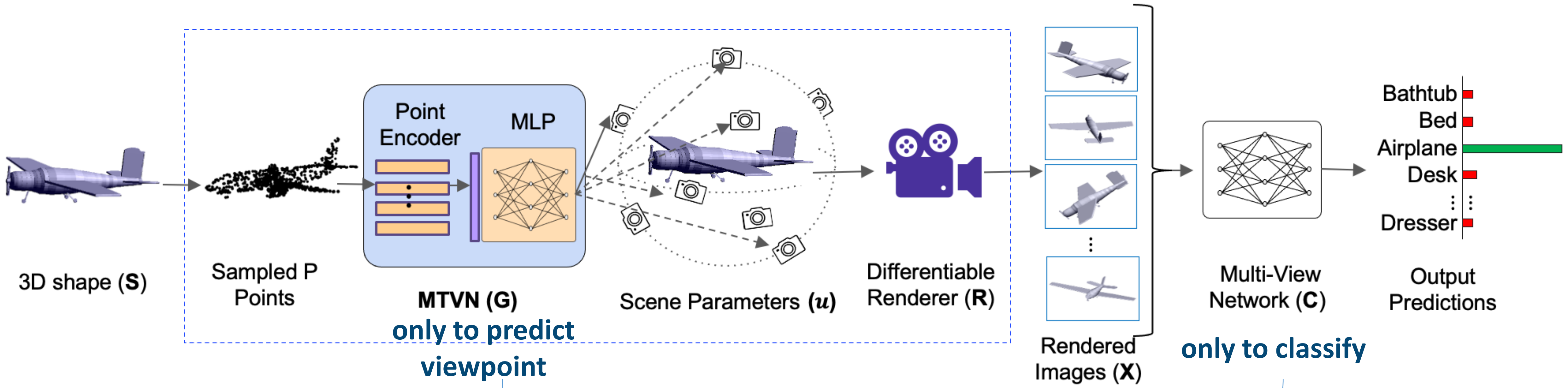
Research Progress

2022.03.04

Guan Yunyi

Review - MVTN



only to predict viewpoint

only to classify

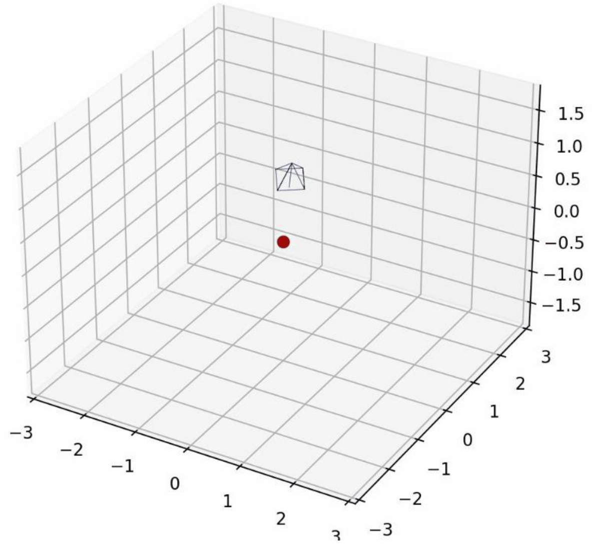
C and G are trained jointly on the same loss

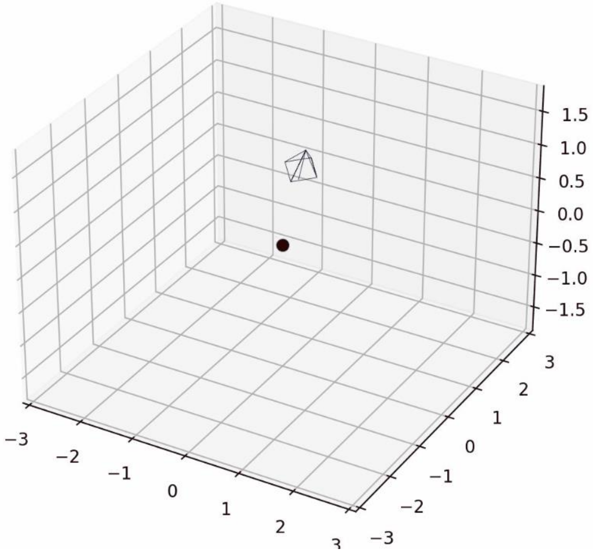
$$\underset{\boldsymbol{\theta}_{\mathbf{C}},\boldsymbol{\theta}_{\mathbf{G}}}{\arg\min} \sum_{n}^{N} L\left(\mathbf{C}(\mathbf{R}(\mathbf{S}_n,\mathbf{u}_n))\ ,\ y_n\right),$$

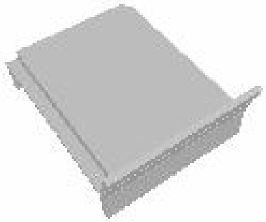
$$\text{s. t.}\quad \mathbf{u}_n = \mathbf{u}_{\text{bound}}.\tanh\left(\mathbf{G}(\mathbf{S}_n)\right)$$

Review – Results of `learned_circular`, `nb_views=1`

- Only moves within a small range of the initial viewpoint (essentially <u>in XZ plane)</u>

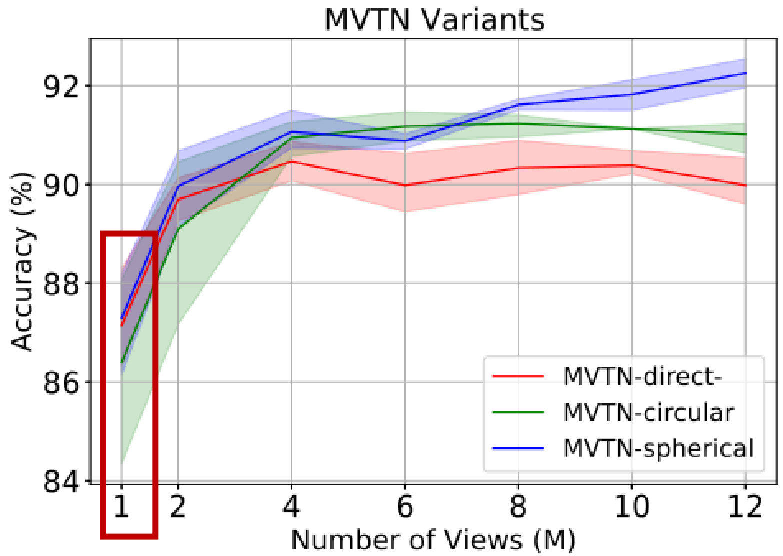






Epoch=100:

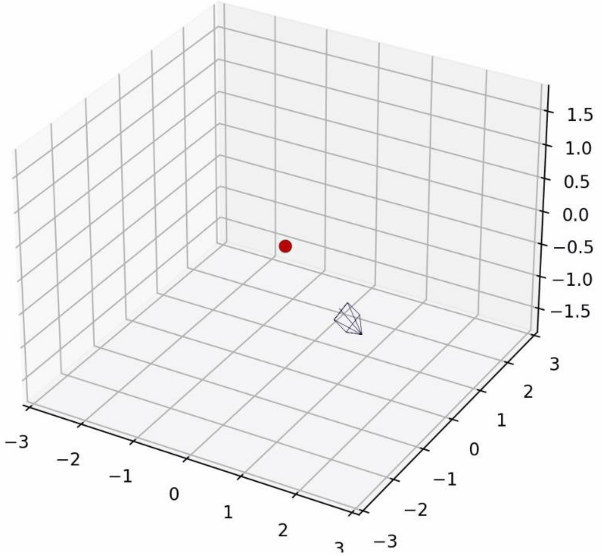
- train acc: 80.29, train loss: 0.6921
- val acc: 72.37, val Loss: 1.0615
- Current best val acc: **73.01 too low**

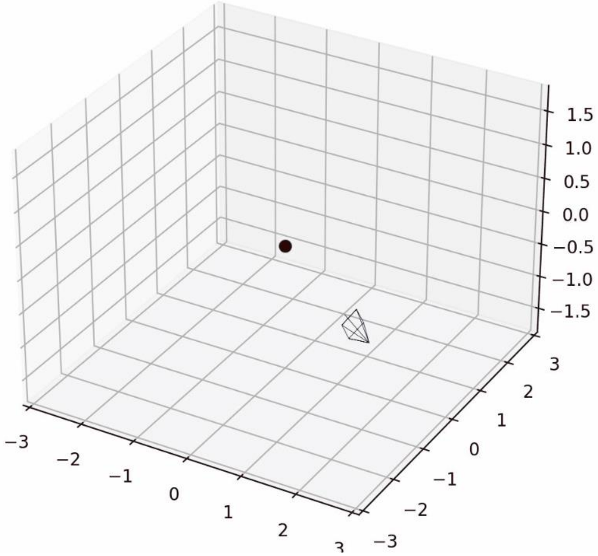


should be around 84-88

Review – Results of `learned_spherical`, `nb_views=1`

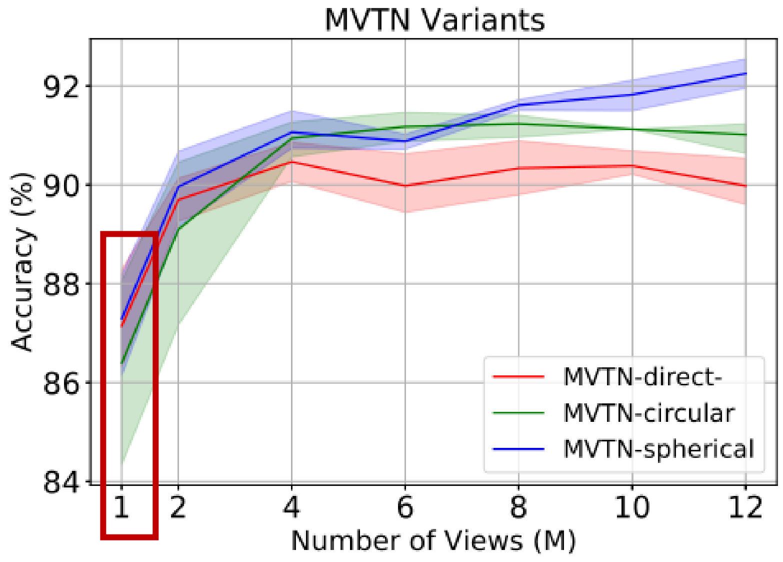
- Only moves within a small range of the initial viewpoint (essentially <u>in XZ plane)</u>





Epoch=100:

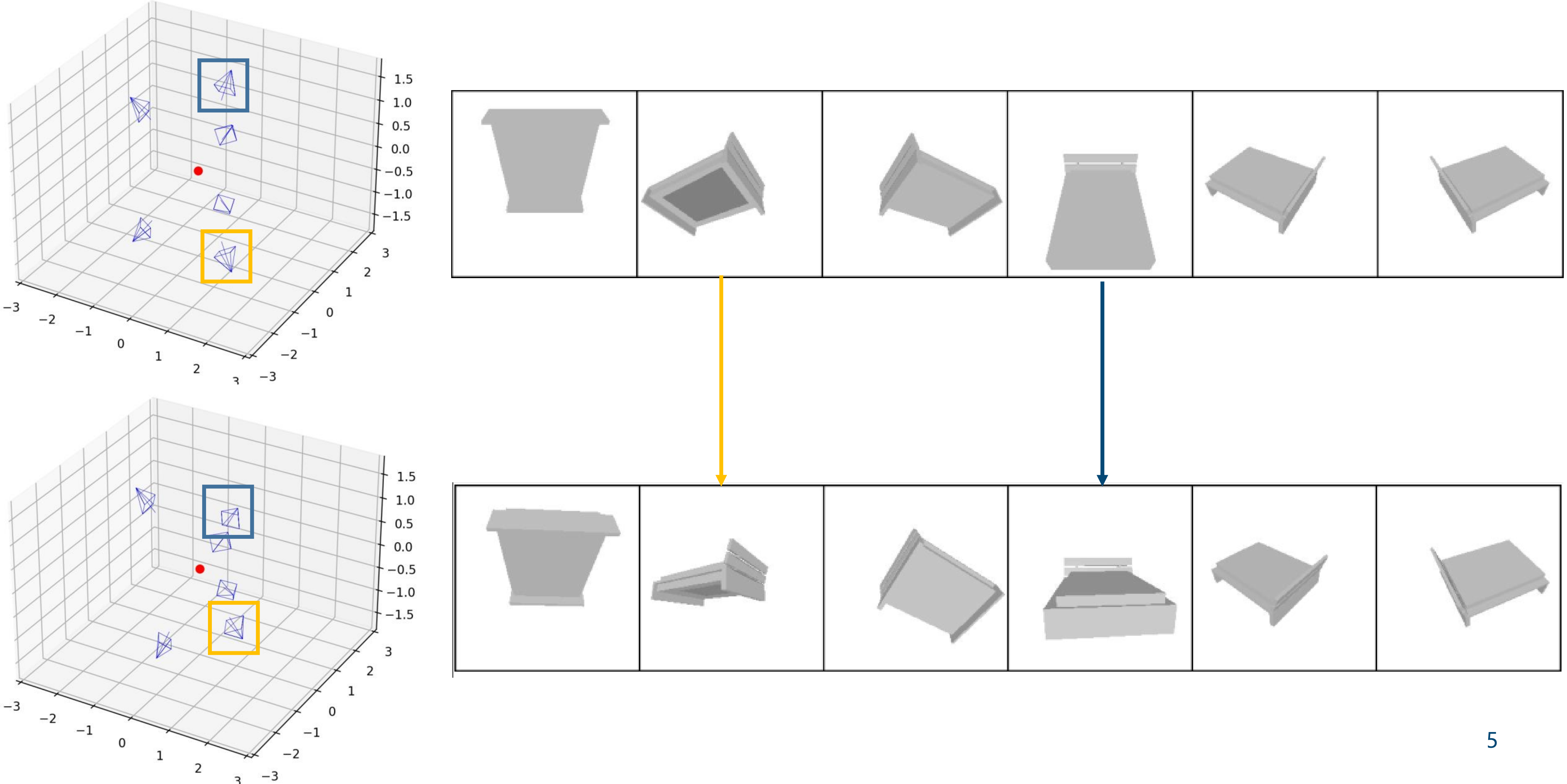
- train acc: 83.62 - train Loss: 0.5915
- val acc: 70.26 - val Loss: 1.1637
- Current best val acc: **71.96 too low**



should be around 84-88

Review – Paper's result of `learned_spherical`, `nb_views=6`

- When there are multiple viewpoints, the range of movement is larger



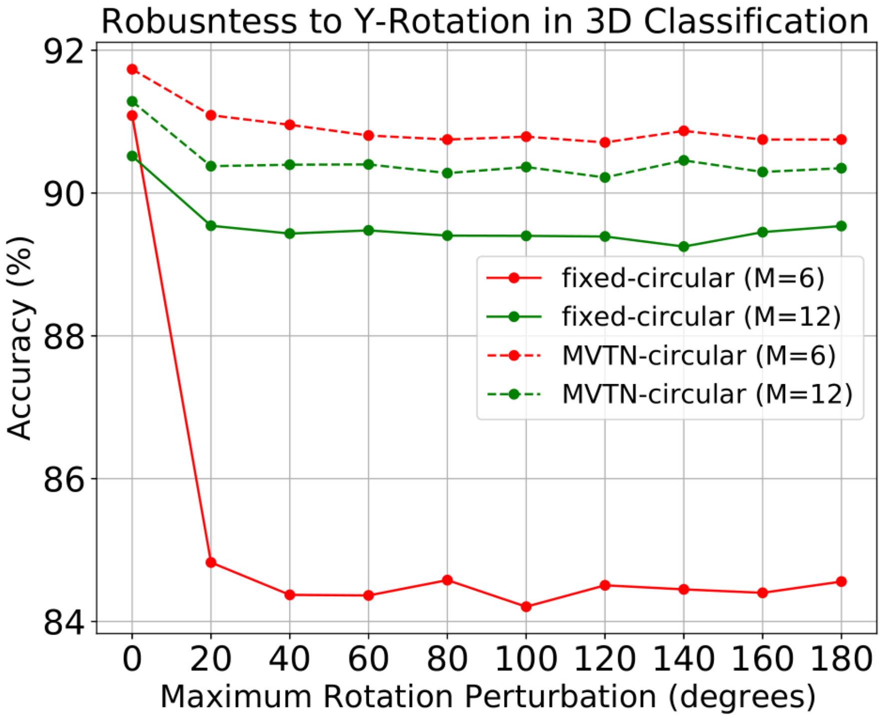
What I did – Test with rotation

- There is a test for rotation robustness in the paper:

(1) train models on `ModelNet40`

(2) randomly rotate objects around <u>Y-axis</u> with different angles

(3) study the effect of varying rotation on the classification accuracies



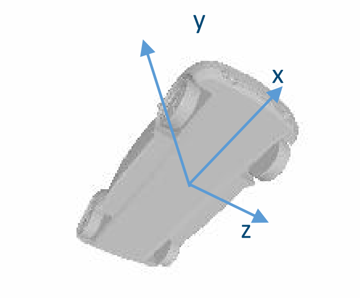
⇒ **Conclusion:** MVTN maintains high accuracy regardless of the degrees of rotation around Y-axis

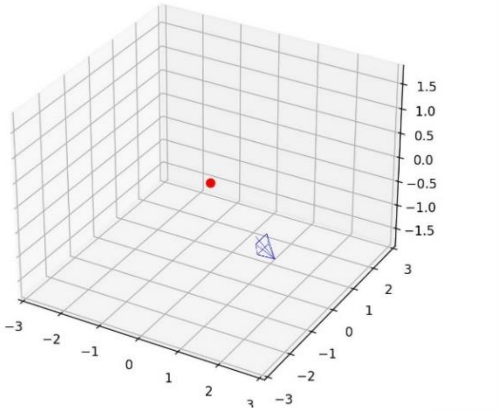
- What about **the movement of cameras**?
- What about **X-axis** and **Z-axis**?
- What about **with only 1 viewpoint**?

What I did – Test with rotation `trained 100 epochs`, `learned_sphrical`, `nb_views=1`

- **rotate around X,Y and Z axis** and plot the results for 3 times
- Angle of rotation for each time: random from [-180°, 180°]

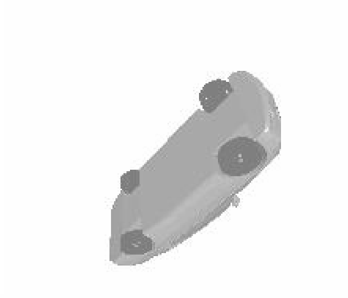
Test without rotation

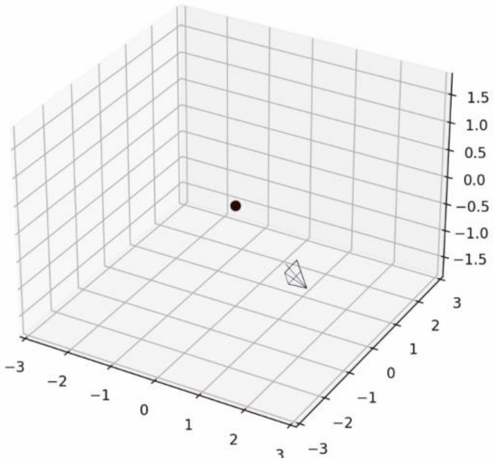




Val Acc: 70.50

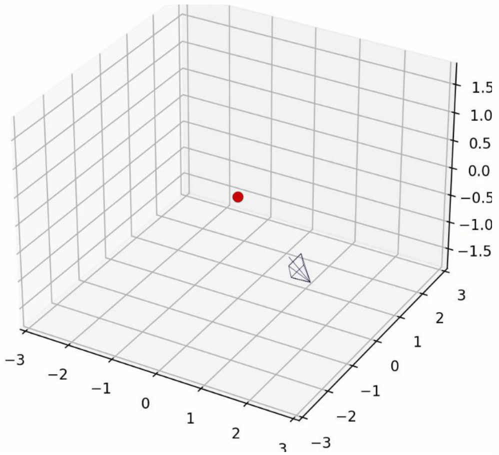
X-axis





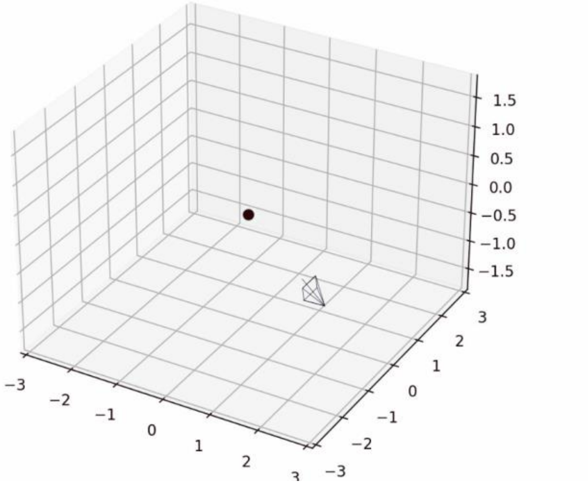
Val Acc: **24.37**

Y-axis



Val Acc: 60.14

Z-axis

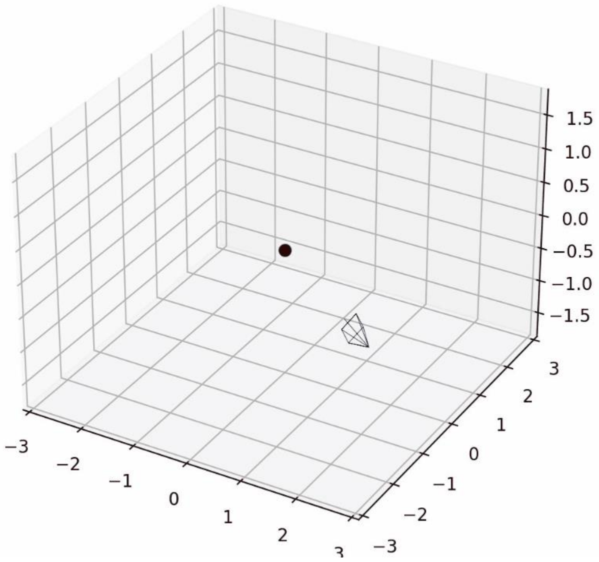


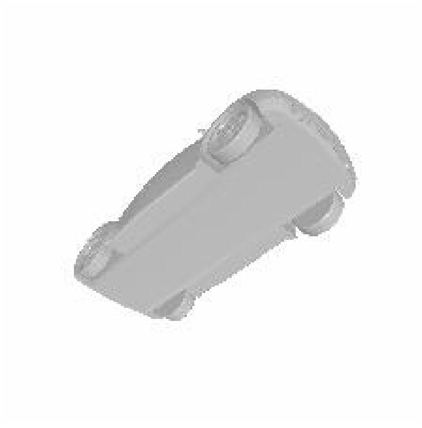
Val Acc: **27.77**

What I did - Conclusion

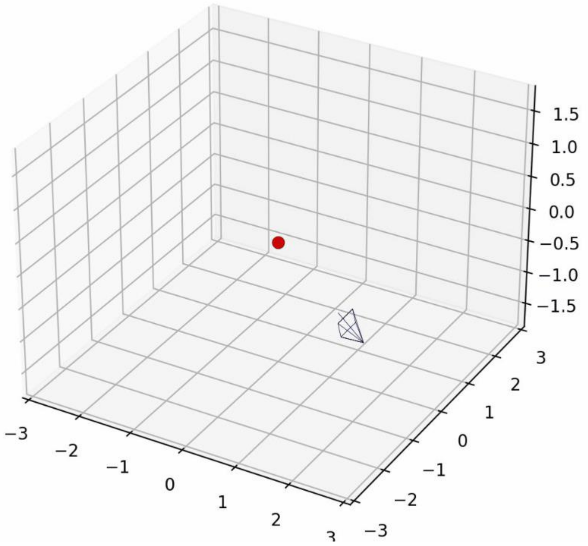
`learned_sphrical`, `nb_views=1`

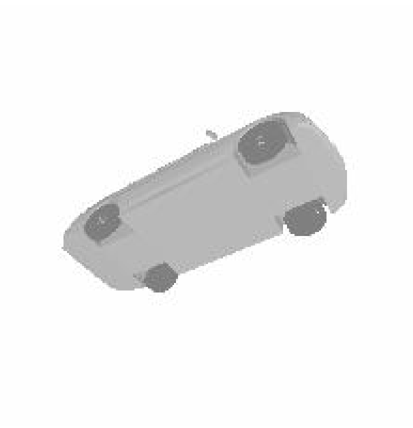
training





Y-axis rotation





- Perhaps, **with `nb_view=1`, MVTN can only learn to move the viewpoint in the XZ plane**, which explains its robustness to Y-axis rotation
- Can this be used as a <u>comparison</u>? Although MVTN knows the overall shape of the object, it still cannot learn how to find the NBV.

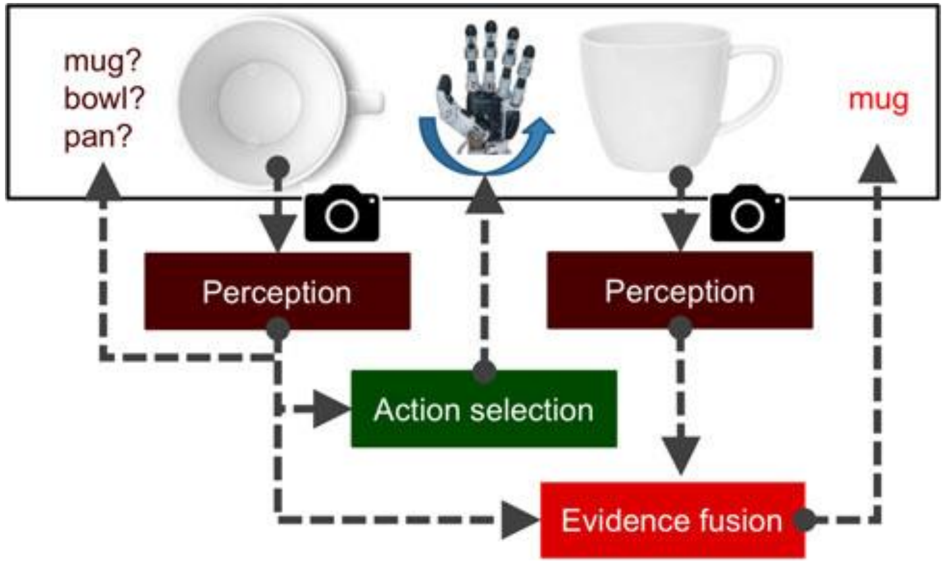
  -> without RL,

  it is hard to find the NBV

Maybe because the training epoch is too small, I will try to increase the epoch and continue testing

Standard framework of Active Object Recognition(AOR)

3 main modules



Given an initial viewpoint and a set of possible actions:

(1) **Action selection**: learns action policies -> finds the NBV

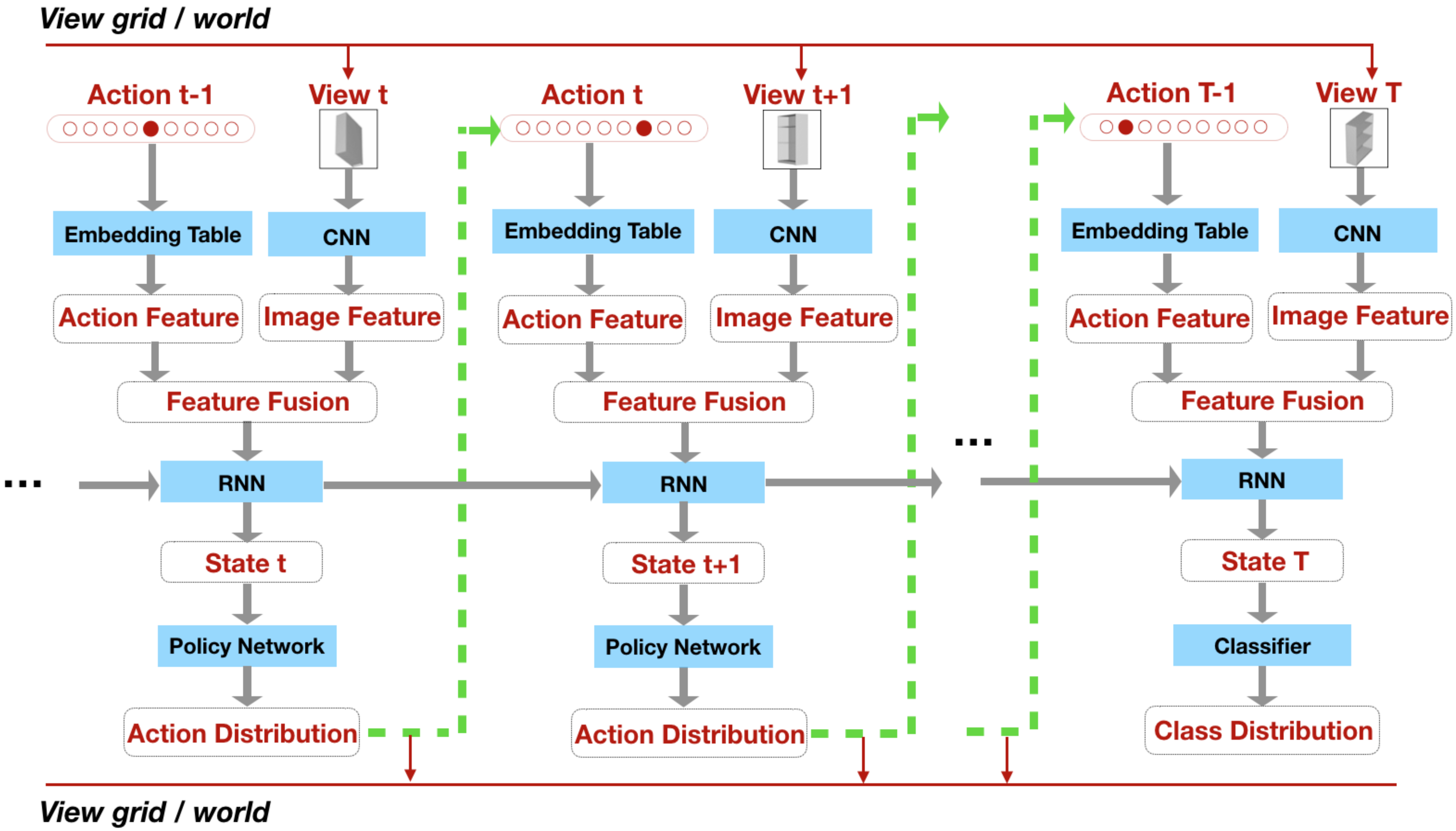
(2) **Perception**: processes input per timestep -> prepare and recognition

(3) **Evidence fusion**: aggregates visual and action information, RNN

Jointly trained

An example pipeline of AOR system

Wei, Wei & Yu, Haonan & Zhang, Haichao & Xu, Wei & Wu, Ying. (2021). MetaView: Few-shot Active Object Recognition.

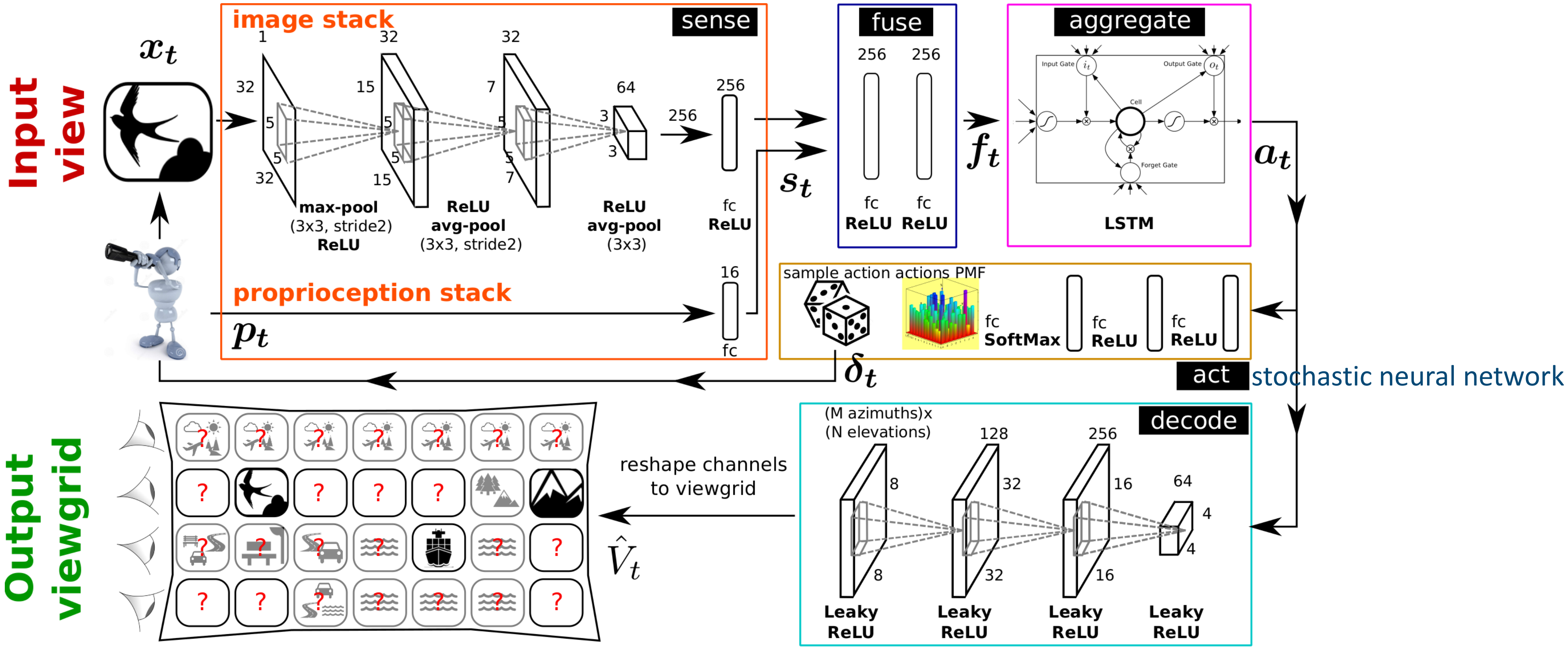


Papers of active vision

D. Jayaraman and K. Grauman:

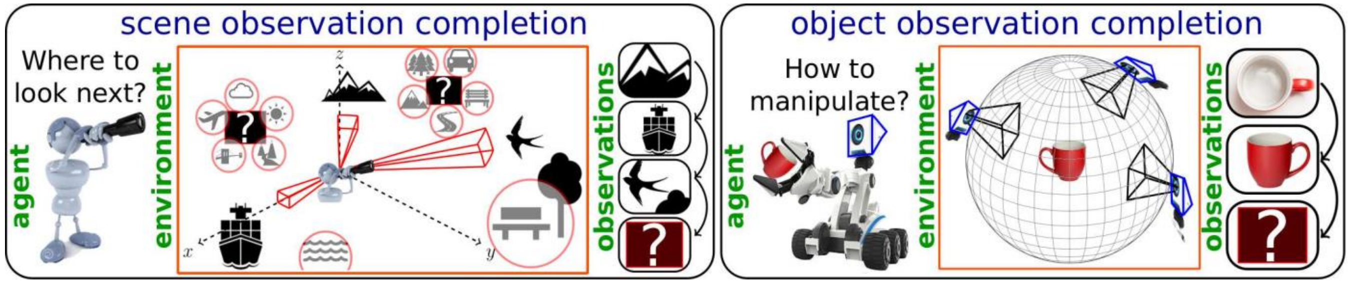
- Learning to Look Around: Intelligently Exploring Unseen Environments for Unknown Tasks (CVPR2018)

Architecture



To **minimize loss**:
stochastic
gradient decent
+
REINFORENCE

2 settings



Papers of active vision

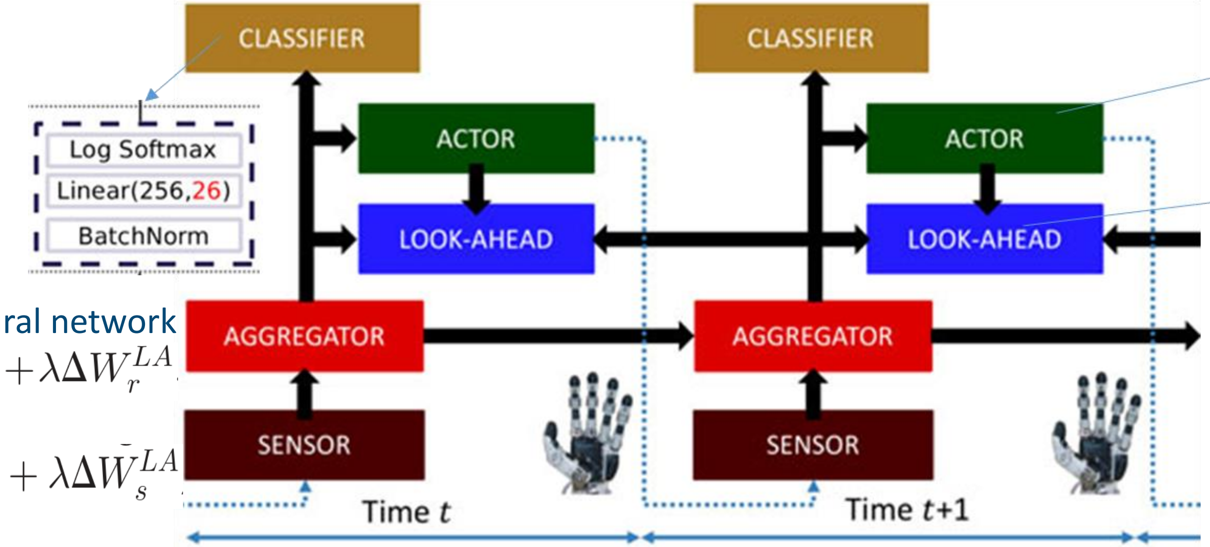
D. Jayaraman and K. Grauman:

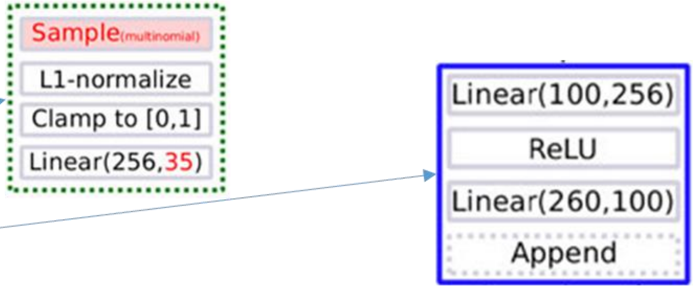
- End-to-End Policy Learning for Active Visual Categorization (2019)

$\Delta W_a = \Delta W_a^{RL},$

stochastic neural network

$\Delta W_c = \Delta W_c^{SM}.$





recurrent neural network

$\Delta W_r = \Delta W_r^{RL} + \Delta W_r^{SM} + \lambda \Delta W_r^{LA}$

$\Delta W_s = \Delta W_s^{RL} + \Delta W_s^{SM} + \lambda \Delta W_s^{LA}$

network weights $W = [W_c, W_a, W_s, W_r]$

are **trained jointly at time t**

additional module, **LOOK-AHEAD**: -> estimating the effects of active motions

- Predicts current timestep features $\hat{\boldsymbol{a}}_t = \text{LOOKAHEAD}(\boldsymbol{a}_{t-1}, \boldsymbol{m}_t)$
- Adds a part to training gradients $\Delta W_{\backslash ca}^{LA} = \sum_{i=1}^{N} \sum_{t=2}^{T} \nabla_{W_{\backslash ca}} d(\hat{\boldsymbol{a}}_t, \boldsymbol{a}_t | \boldsymbol{a}_{t-1}, \boldsymbol{m}_t)$

Papers of active vision - Problems

2. Active view-changing part

(1) **How** to find the **Next-Best-View**?

(2) **What time** to **stop** view-changing and **output** the result?

- No option of **autonomous terminating**

- Whether is necessary to start changing the viewpoint

  -> how to determine if the current viewpoint is good enough?

- What time to stop the view-changing efficiently

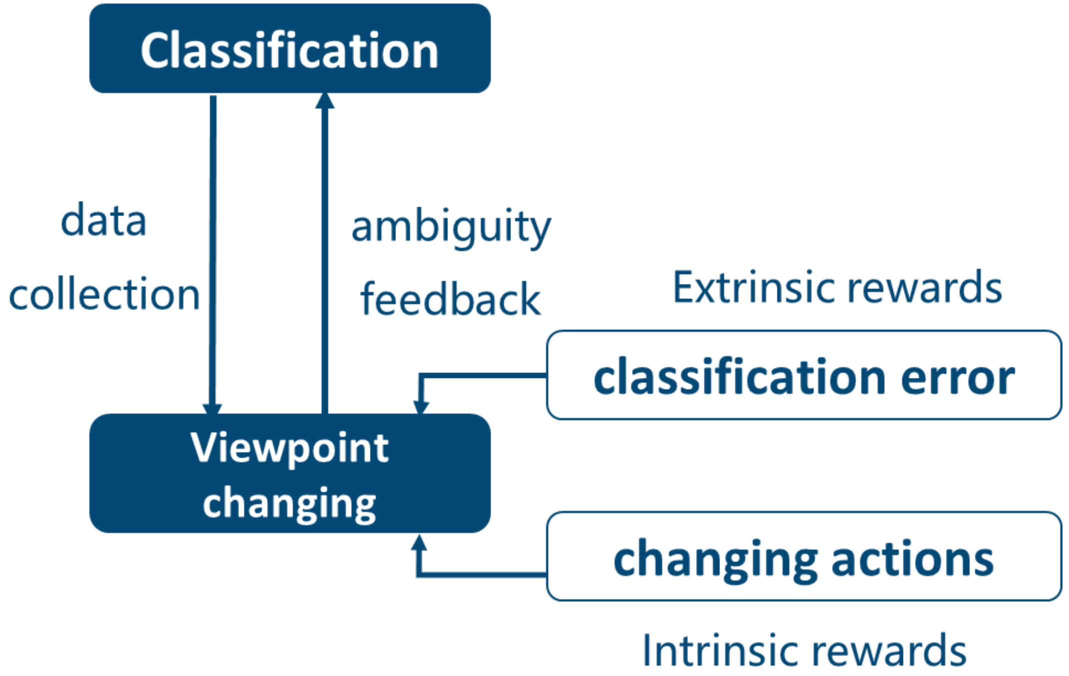
  -> how to minimize the number of timesteps?

- Not combined with **pose estimation**

  (strength of RotationNet, however no idea now…)

Summary of possible ideas

- Focus on active object recognition
- While training:



Choose the next action based on **KL divergence** to **reduce ambiguity**

- While testing:

Input: initial viewpoint and image

1.Predict the category by RotationNet

2.Determine whether to change viewpoint:

If yes,

predict the NBV,

go to step1 (accuracy should ↑).

If not,

output classification result and time.

Next to do

- Coding:

- Continue to test the rotation of MVTN with larger epoch

- Find the source code of LOOKAHEAD and try to run it

- Paper reading:

- Continue to read the paper of LOOKAHEAD carefully

- Find some new papers of AOR

Additional discussion

- Whether the `ModelNet40` dataset used in MVTN aligned?

- Original text from MVTN supplemental material:

2.2. Rotation Robustness

A common practice in the literature in 3D shape classification is to test the robustness of models trained on the aligned dataset by injecting perturbations during test time [20]. We follow the same setup as [20] by introducing random rotations during test time around the Y-axis (gravity-axis). We

- Original text from [20]:

Robustness to point permutation and rigid transformation. We compare the robustness of our RS-CNN with PointNet [24] and PointNet++ [26]. Note that all the models are trained without related data augmentations, *e.g.*, translation or rotation, to avoid confusion in this test. In addition,

- Is this setup only used for testing rotation robustness?
- I asked the author of MVTN on Github, but haven't gotten an reply yet ......