# Classification confidence via 2D embeddings

Guillem Pascual Guinovart

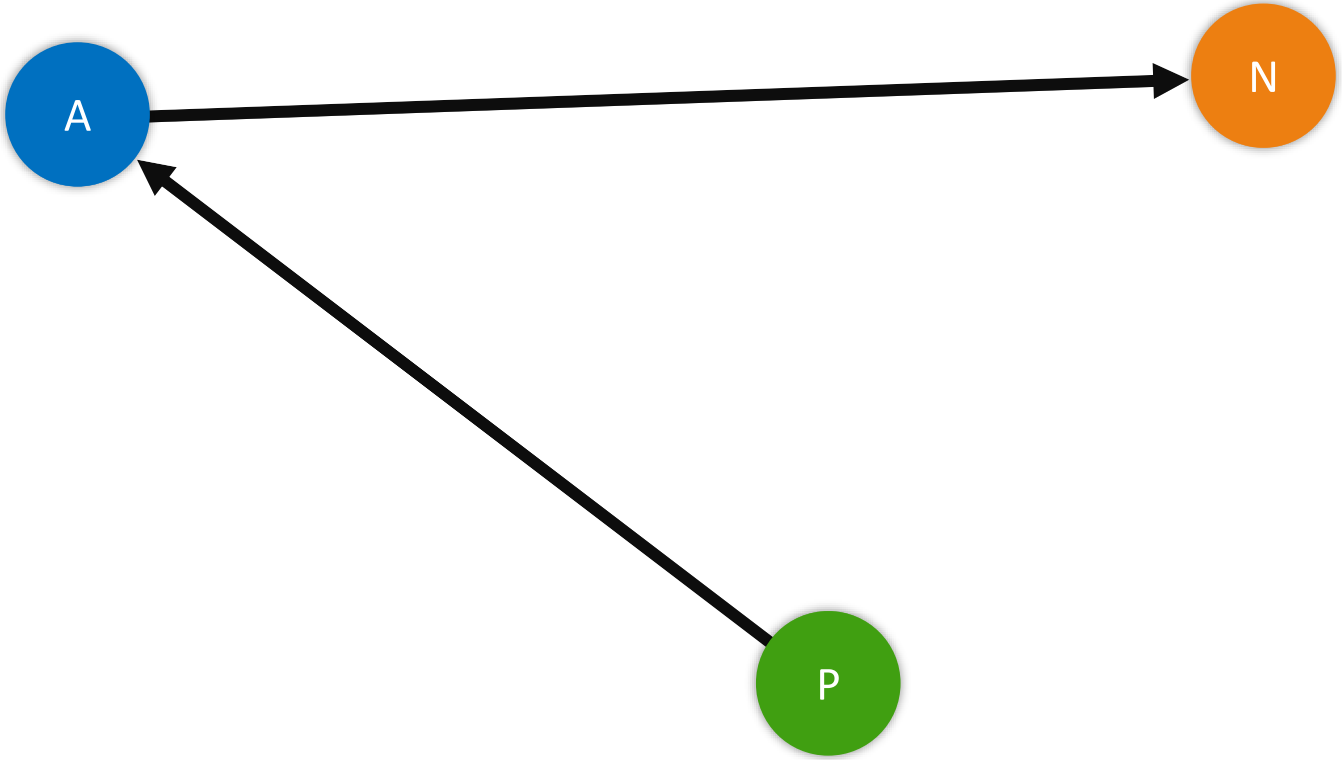
03/12/2020

UNIVERSITAT DE BARCELONA

# MOTIVATION

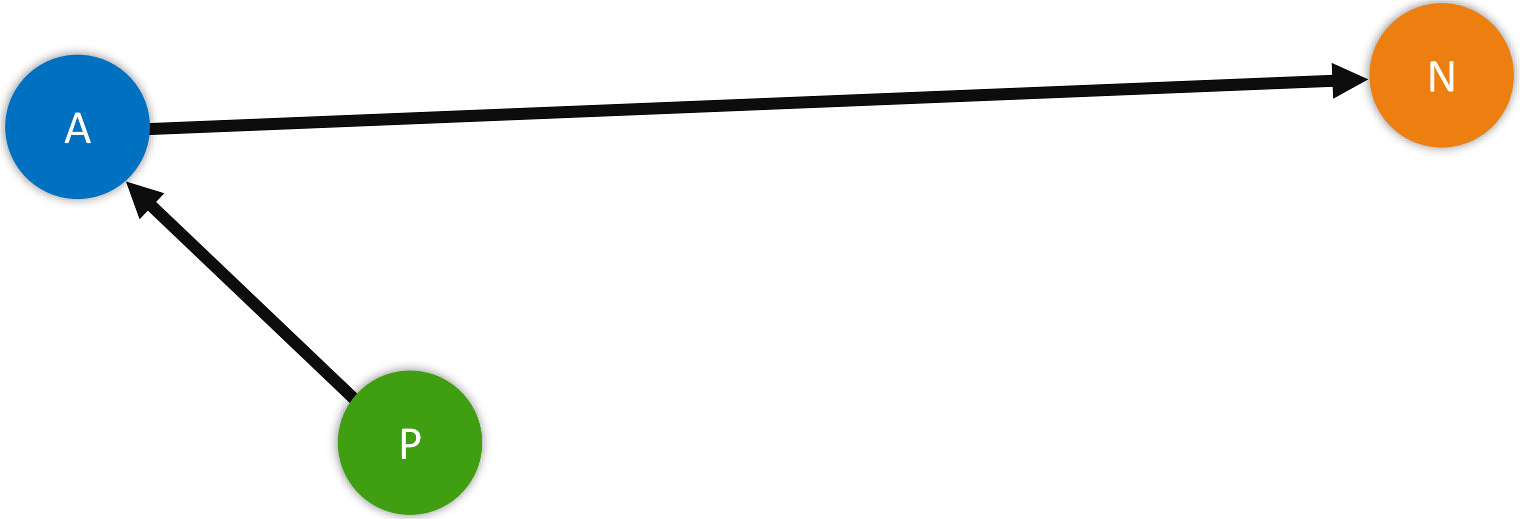
# TRIPLET LOSS 101

Minimize $d(A, P)$
Maximize $d(A, N)$



# TRIPLET LOSS 101

Minimize $d(A,P)$
Maximize $d(A,N)$

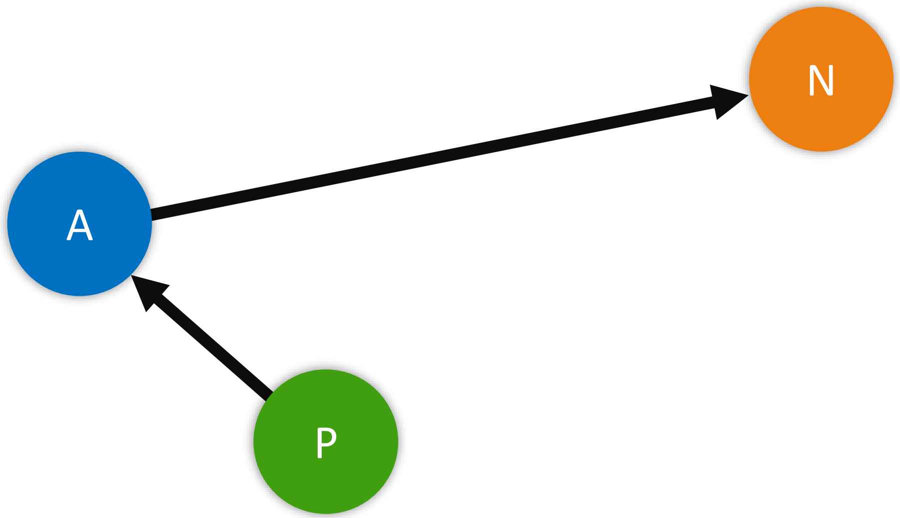



Minimize $max(d(A,P) - d(A,N) + \alpha, 0)$

# TRIPLET LOSS TYPES

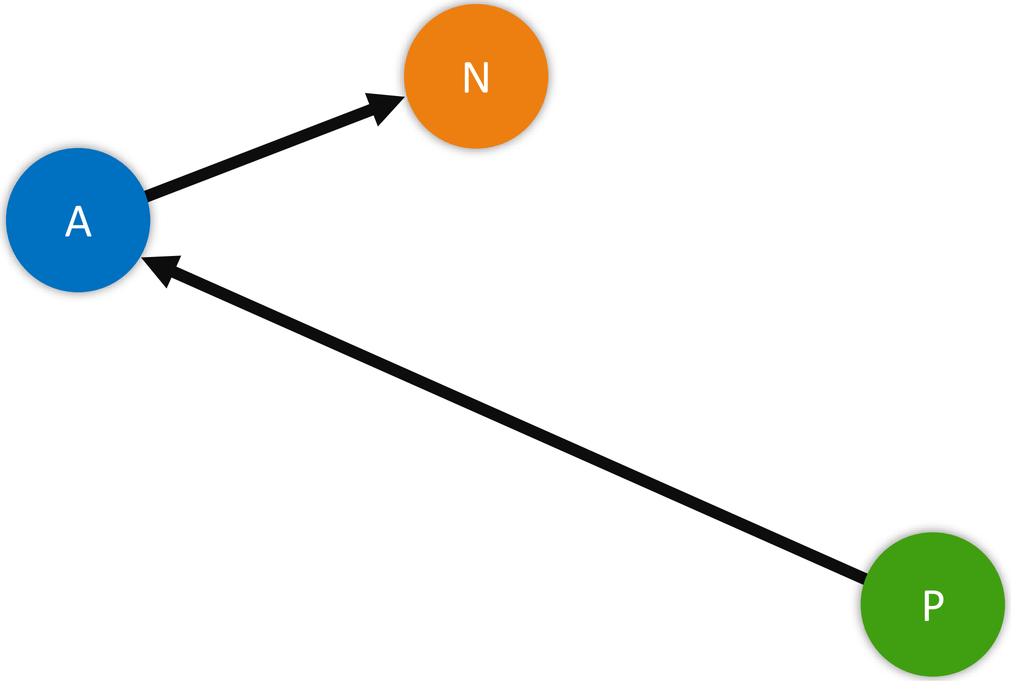
## Easy & Semi-hard Triplets

$$d(A, N) > d(A, P)\ [+\alpha]$$

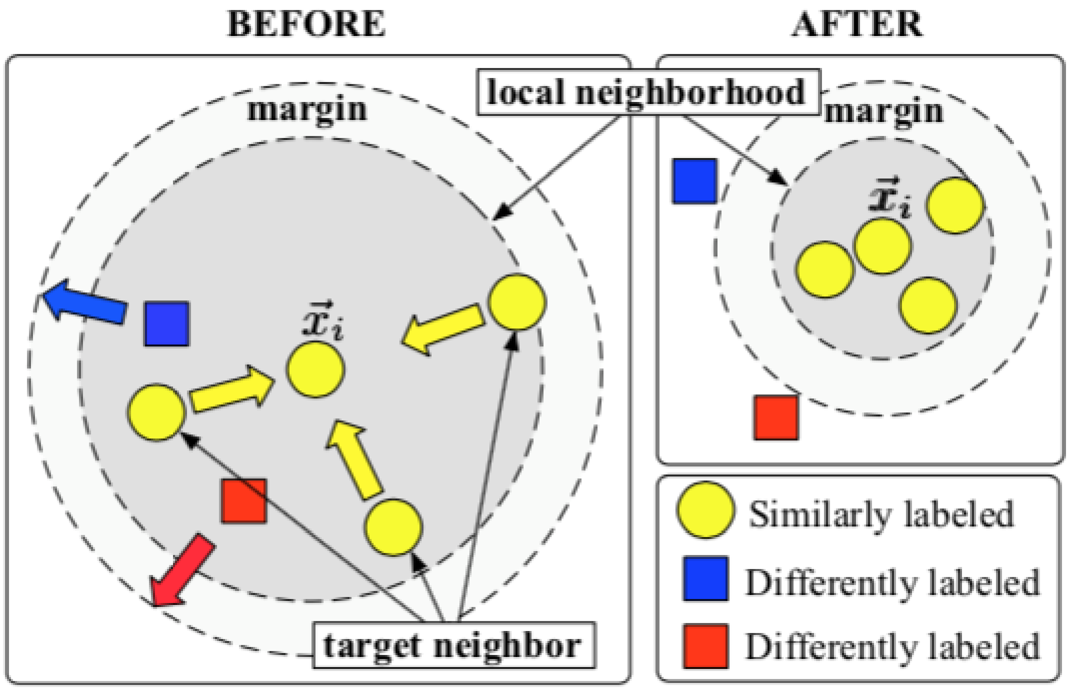



## Hard Triplet

$$d(A, N) < d(A, P)$$



# TRIPLET LOSS TYPES






WEINBERGER, Kilian Q.; SAUL, Lawrence K. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 2009, vol. 10, no 2.

# TRIPLET LOSS DRAWBACKS

- Margin hyper-parameter
  - Low values might cause worse convergence
  - High values can cause an ever-expanding space
- Batch hard or batch all?
  - Sampling strategy?
- Provides no feedback on the embedding

# OBJECTIVES

- Create a confidence measure over embeddings
  - Triplet loss
  - Medical Image
  - OOD
  - Etc.

# N-CENTROIDS LOSS

# NCL: NORMALIZED SPACE

- Problem: High values can cause an ever-expanding space
- Solution: Normalized embedding space

$$E = \frac{E}{\|E\|_2}$$

A $(p-1)$-sphere in $\mathbb{R}^p$

# NCL: METRICS

- Distance metrices $(\|u\| = \|v\| = 1)$
  - Euclidean distance (A)

    $$\|u - v\|_2 = \sqrt{(u - v)^2}$$

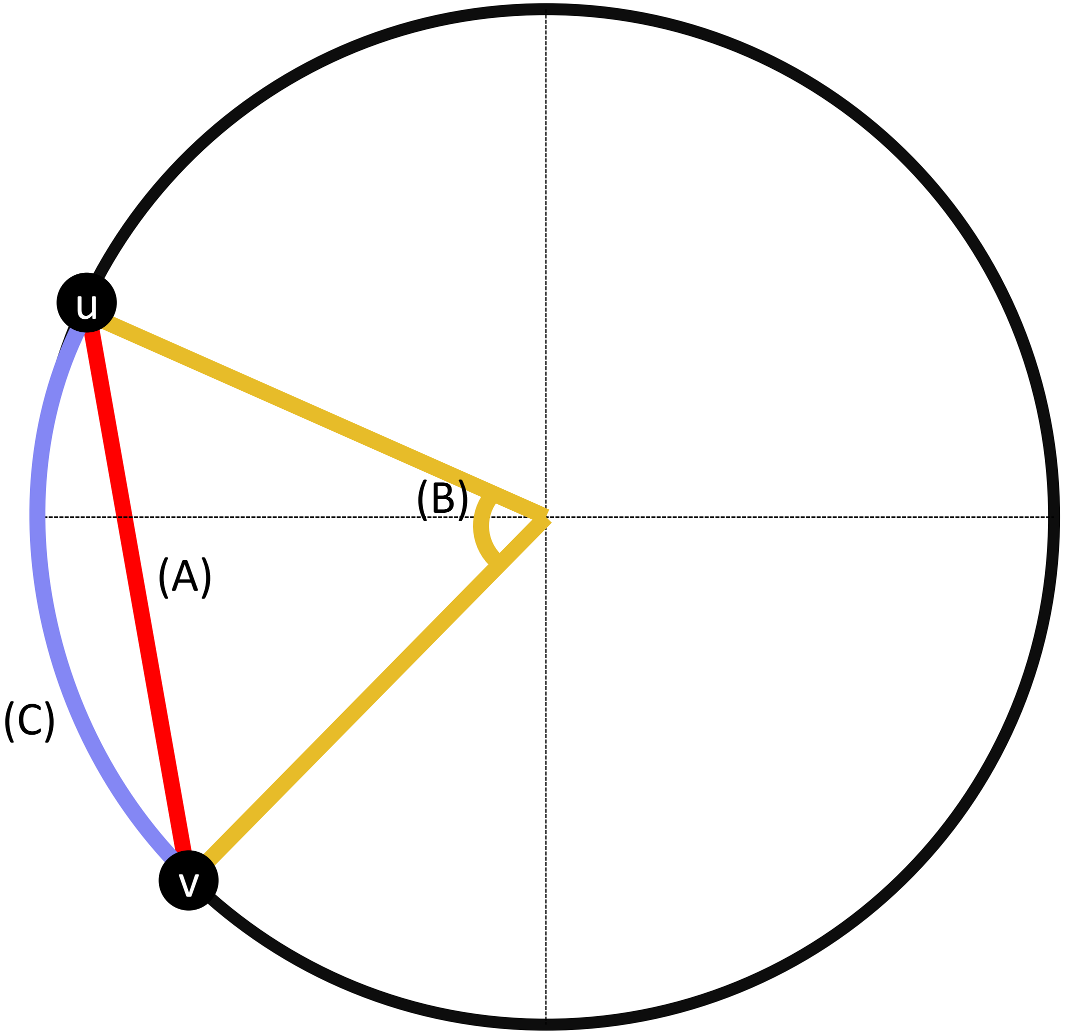
  - Cosine similarity (B)

    $$u \cdot v$$

  - Great-circle distance (C)

    $$2\sin^{-1}(\|u - v\|_2)$$

    Derived of Haversine formula



# NCL: IMPLEMENTATION CONSIDERATIONS

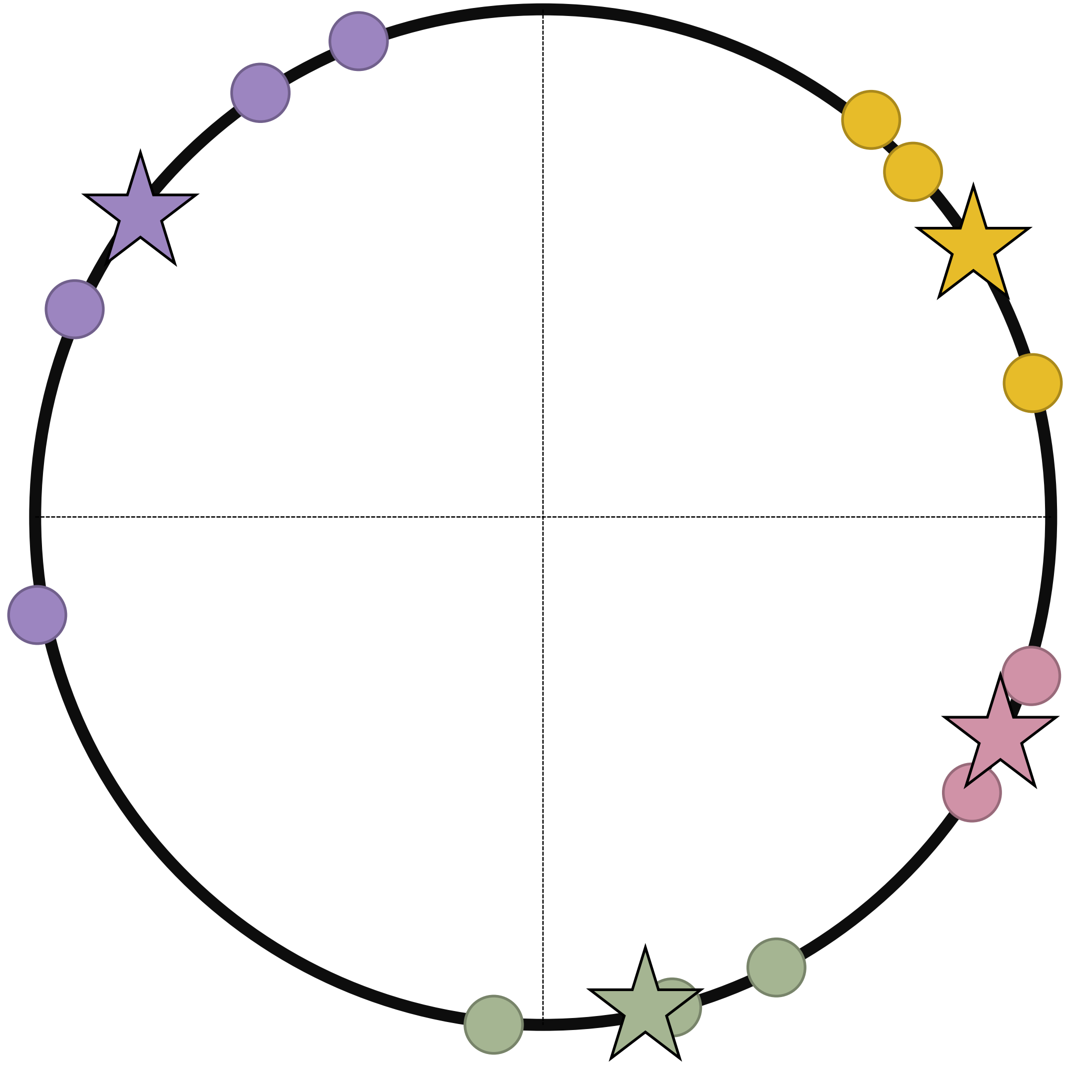
$$sin^{-1}: \quad [-1,1] \qquad\qquad \rightarrow [\pi/2\,, \pi/2]$$
$$\|u - v\|_2 : ([-1,1], [-1,1]) \rightarrow [0,1]$$

But, our friend floating point error makes $\|u - v\|_2 :\ ([-1,1], [-1,1]) \rightarrow (0 - \epsilon, 1 + \epsilon)$

Thus, our implementation must be

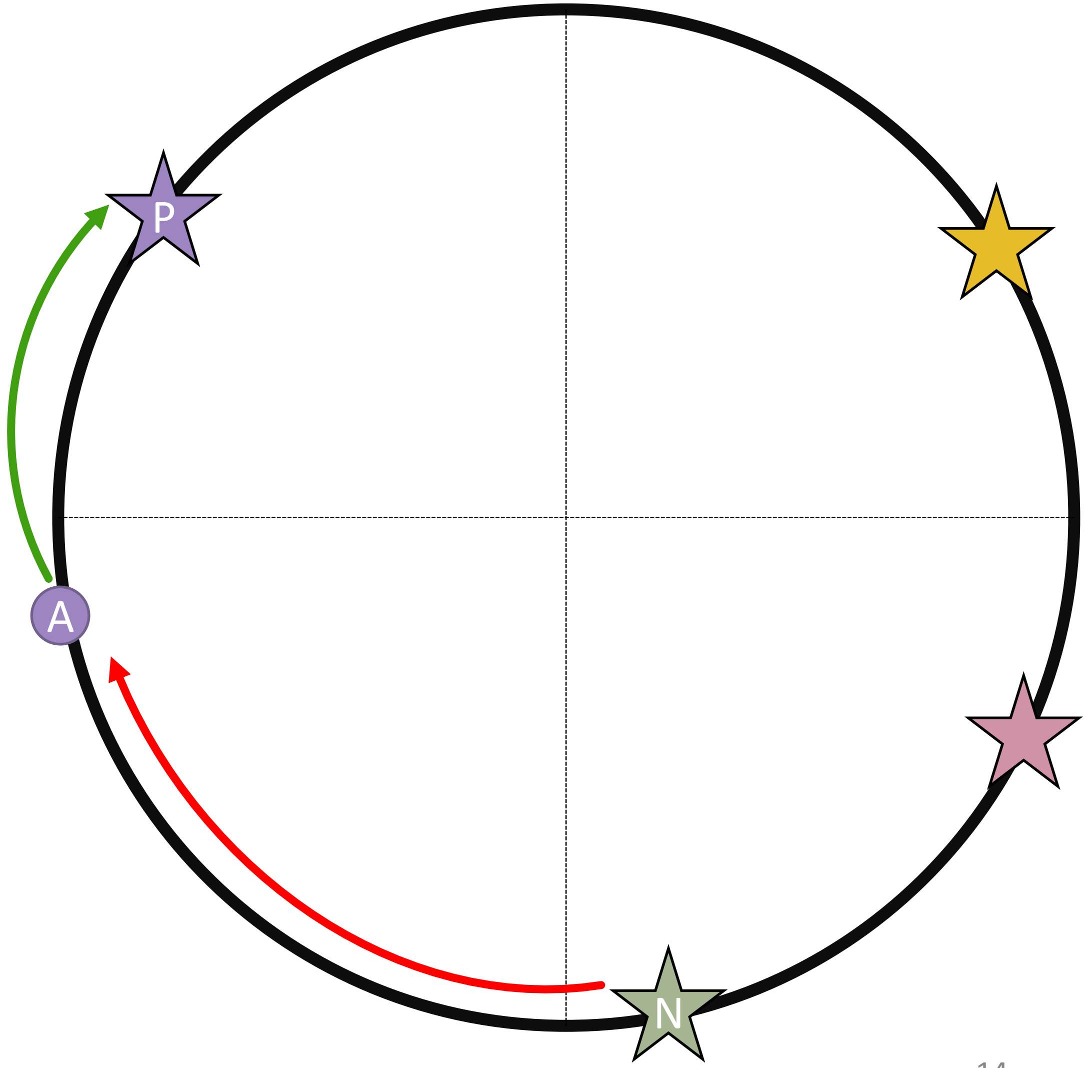
$$2\sin^{-1}(\max(0, \min(1, \|u - v\|_2)))$$

# NCL: STUDY CASE



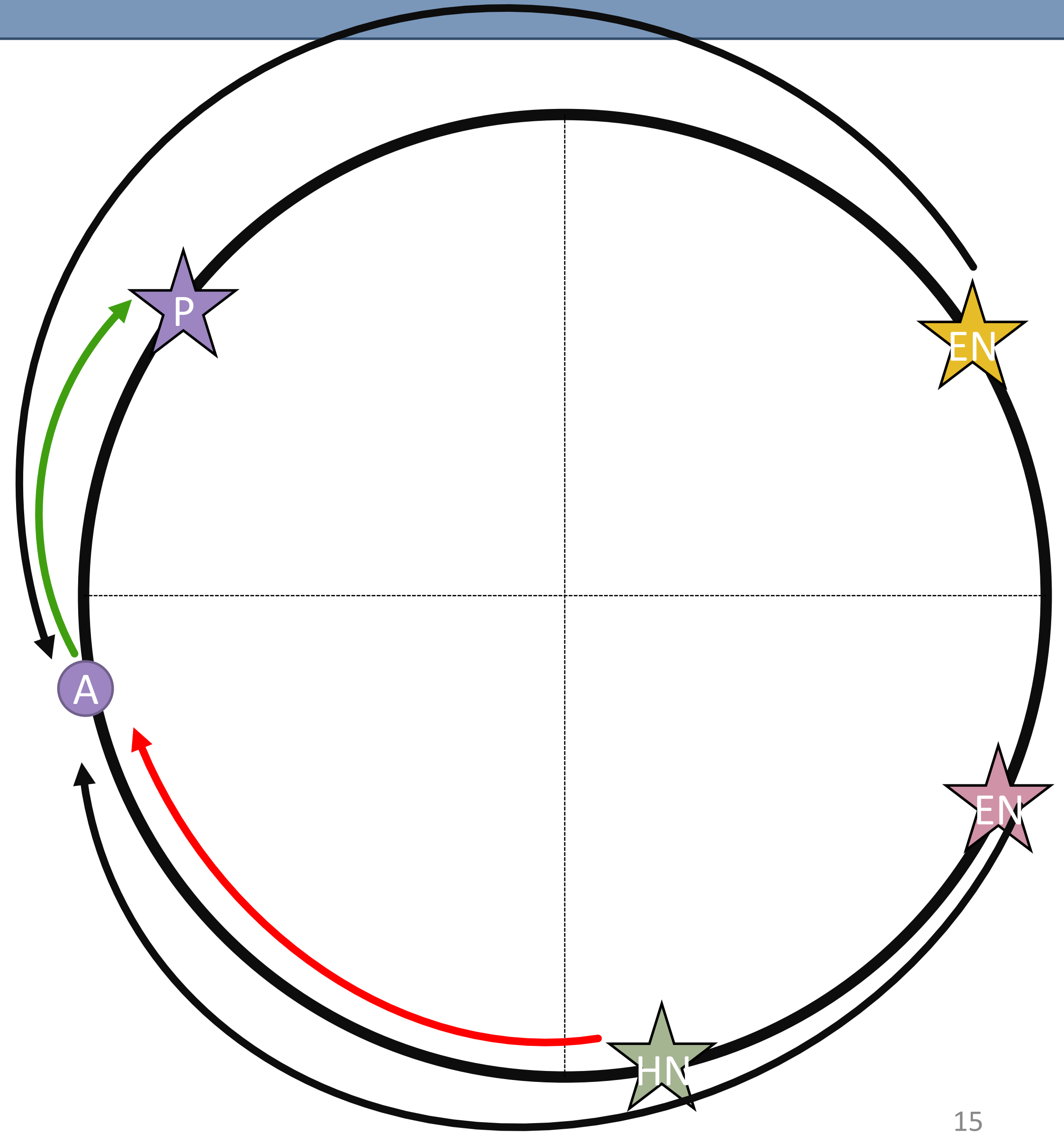
# NCL: CENTROID TRIPLETS

- Problem: Batch all or batch hard
- Solution: Triplets using centroids
  - **d(A, P)**: Minimize distance to own centroid
  - **d(A, N)**: Maximize distance to closest "negative" centroid



# NCL: CENTROID SETS

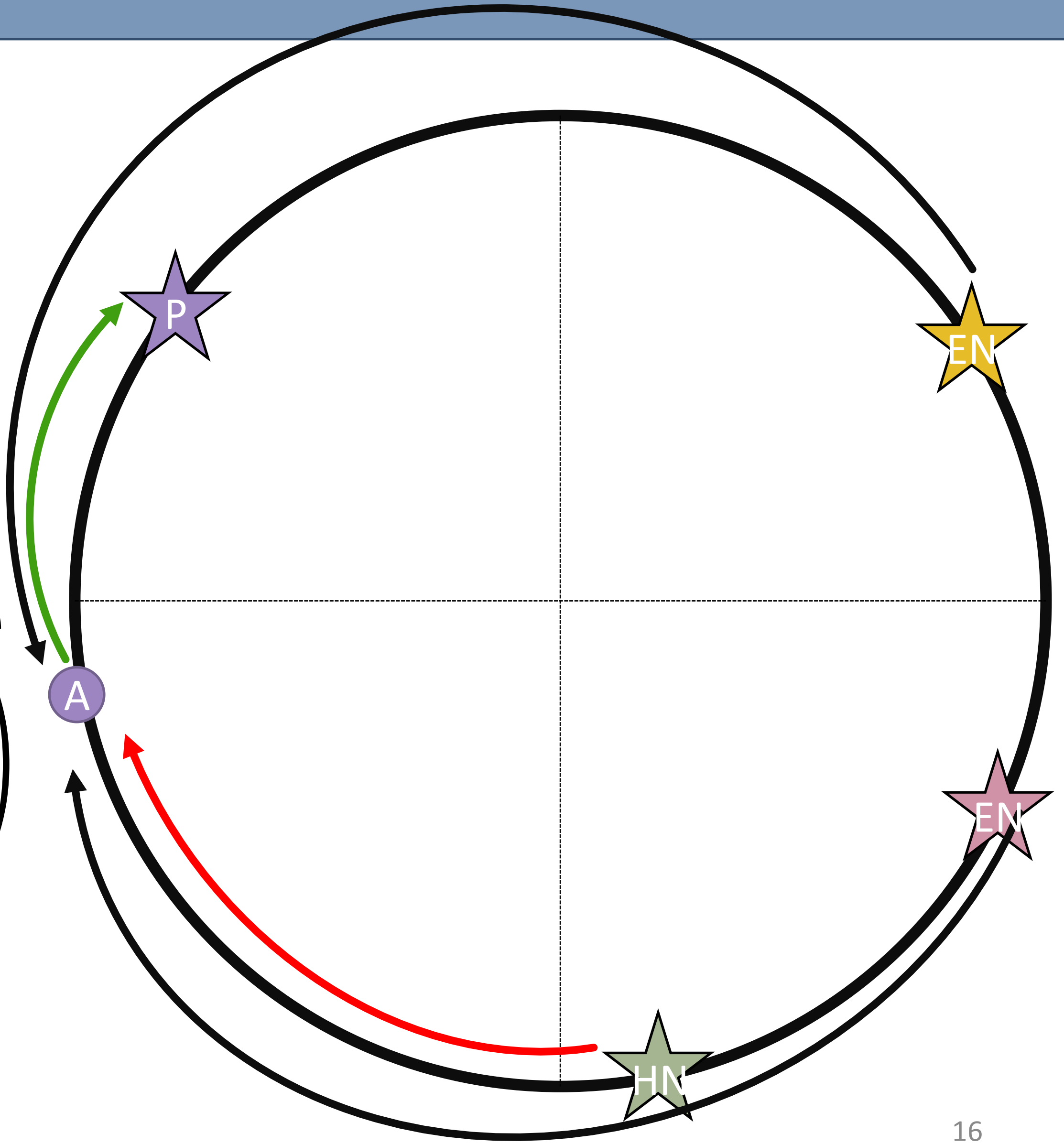
- Problem: Batch all or batch hard
- Solution: Sets using centroids
  - **d(A, P)**: Minimize distance to own centroid
  - **Hard d(A, N)**: Maximize distance to closest “negative” centroid
  - **Easy d(A, N)**: Maximize distance to all other centroids



# NCL: FORMULATION

- Problem: Batch all or batch hard
- Solution: Sets using centroids

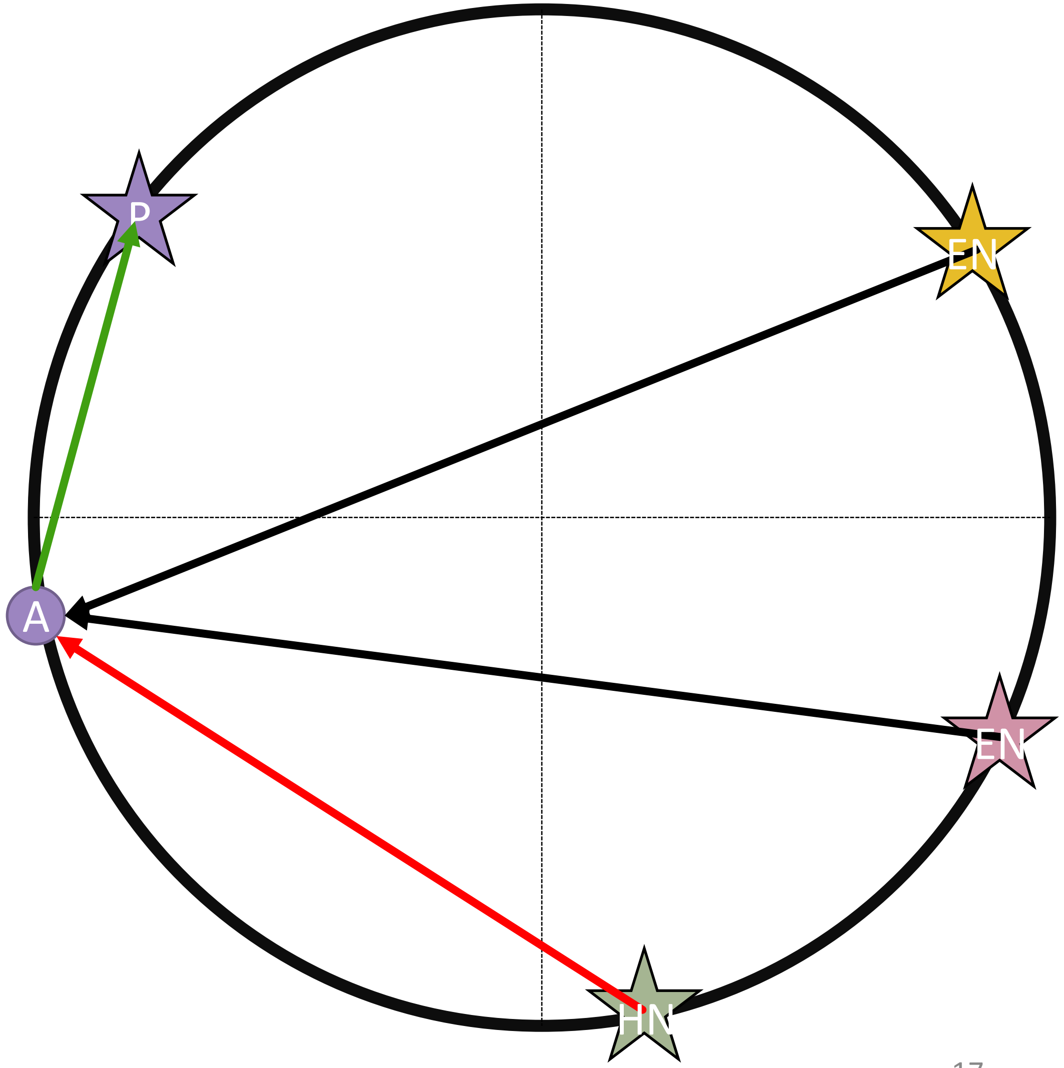
$$\underbrace{\overbrace{d(\star,\bullet)}^{d(A,P)}} - \overbrace{\left( \underbrace{\mathrm{d}(\star,\bullet)}_{\text{Hard}} + \underbrace{\mathrm{K}^{-1} \sum_{i}^{K} d(\star_i \neq \star, \bullet)}_{\text{Easy}} \right)}^{d(A,N)}$$



# NCL: PLEASE HAVE MERCY

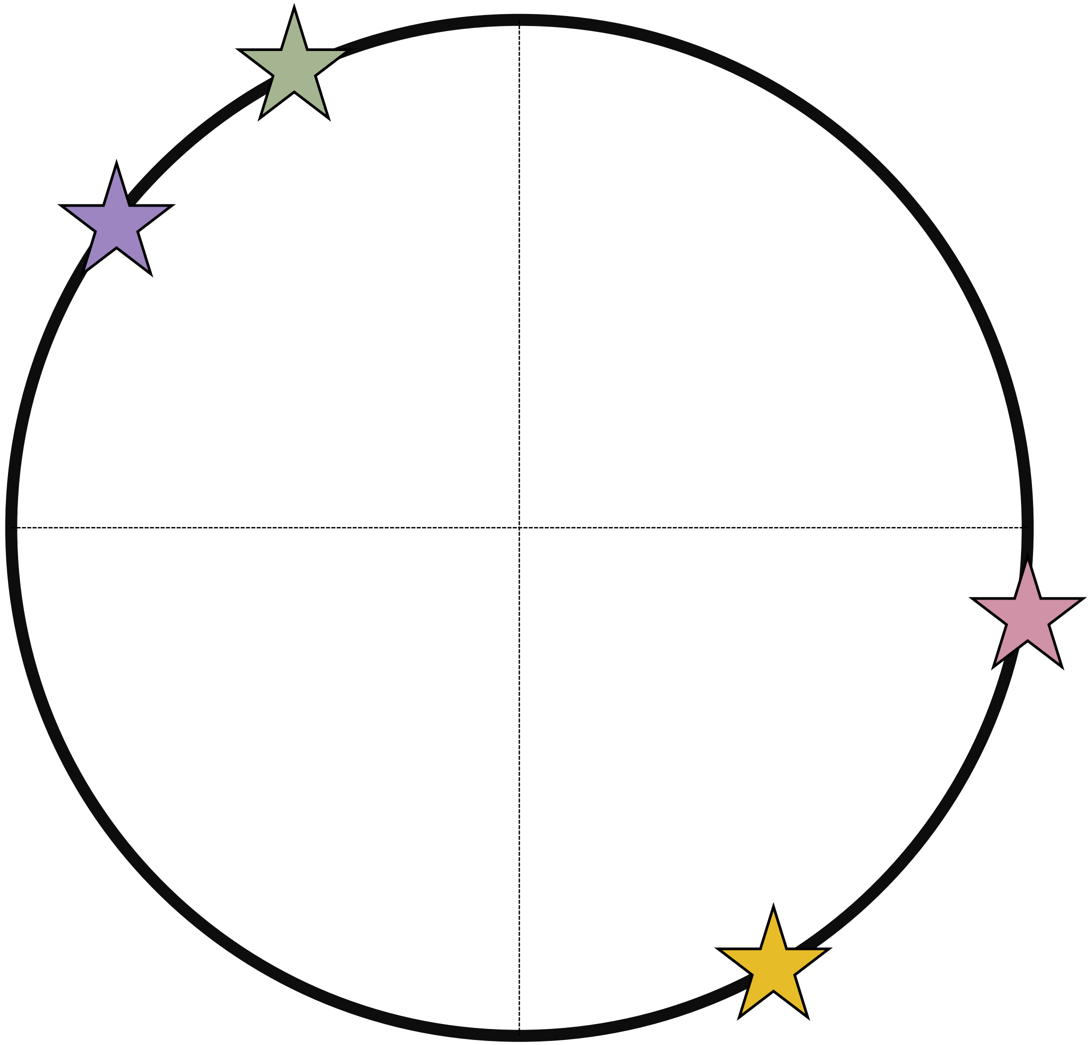
- Problem: Batch all or batch hard
- Solution: Sets using centroids

$$\overbrace{d(\star,\bullet)}^{d(A,P)} - \overbrace{\left(\underbrace{\mathrm{d}(\star,\bullet)}_{\text{Hard}} + \underbrace{\mathrm{K}^{-1}\sum_{i}^{K} d(\star_i \neq \star,\bullet)}_{\text{Easy}}\right)}^{d(A,N)}$$



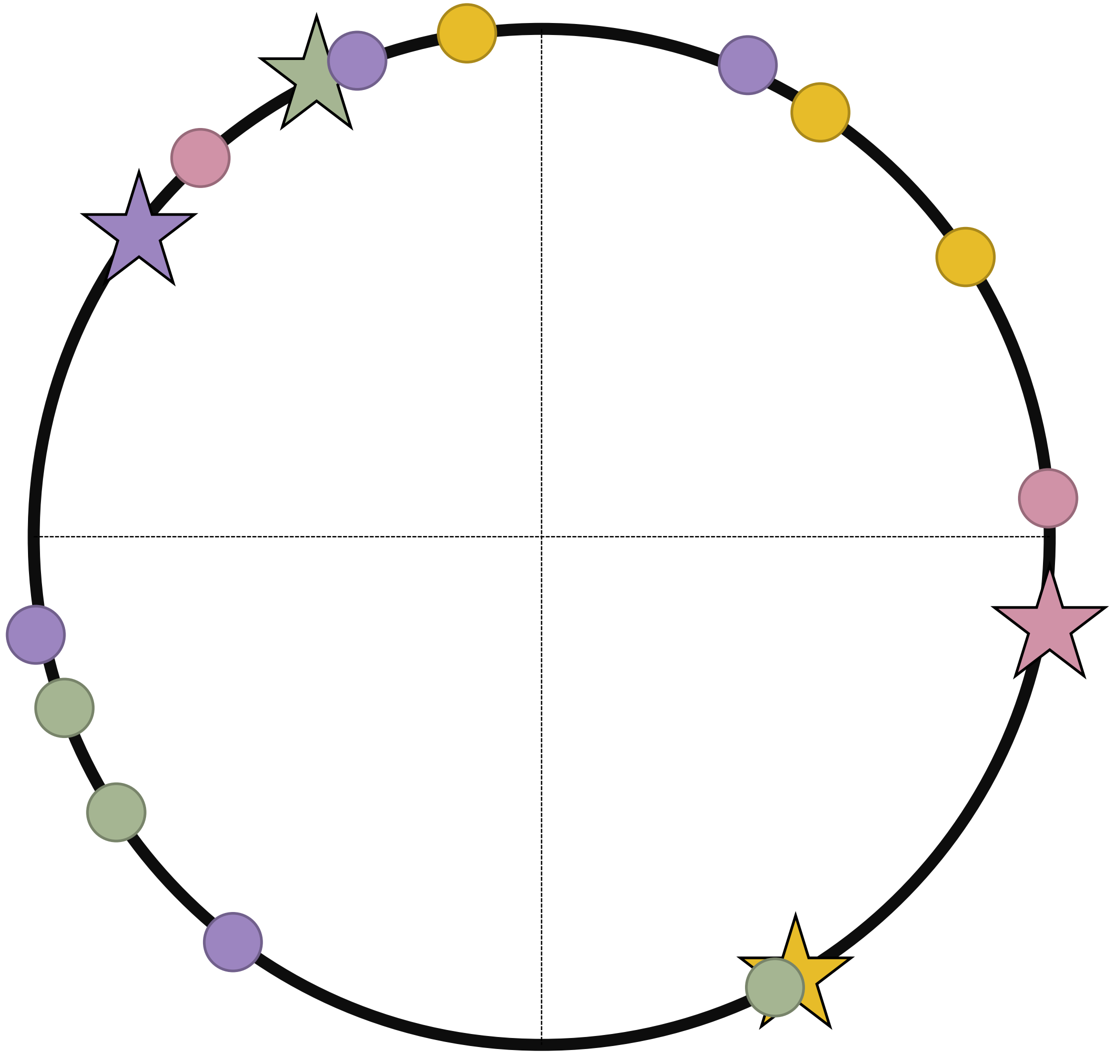
# NCL: FINDING CENTROIDS (1)

1. Choose any random K centroids



# NCL: FINDING CENTROIDS (2)

1. Choose any random K centroids
2. Forward a batch



# NCL: FINDING CENTROIDS (3)

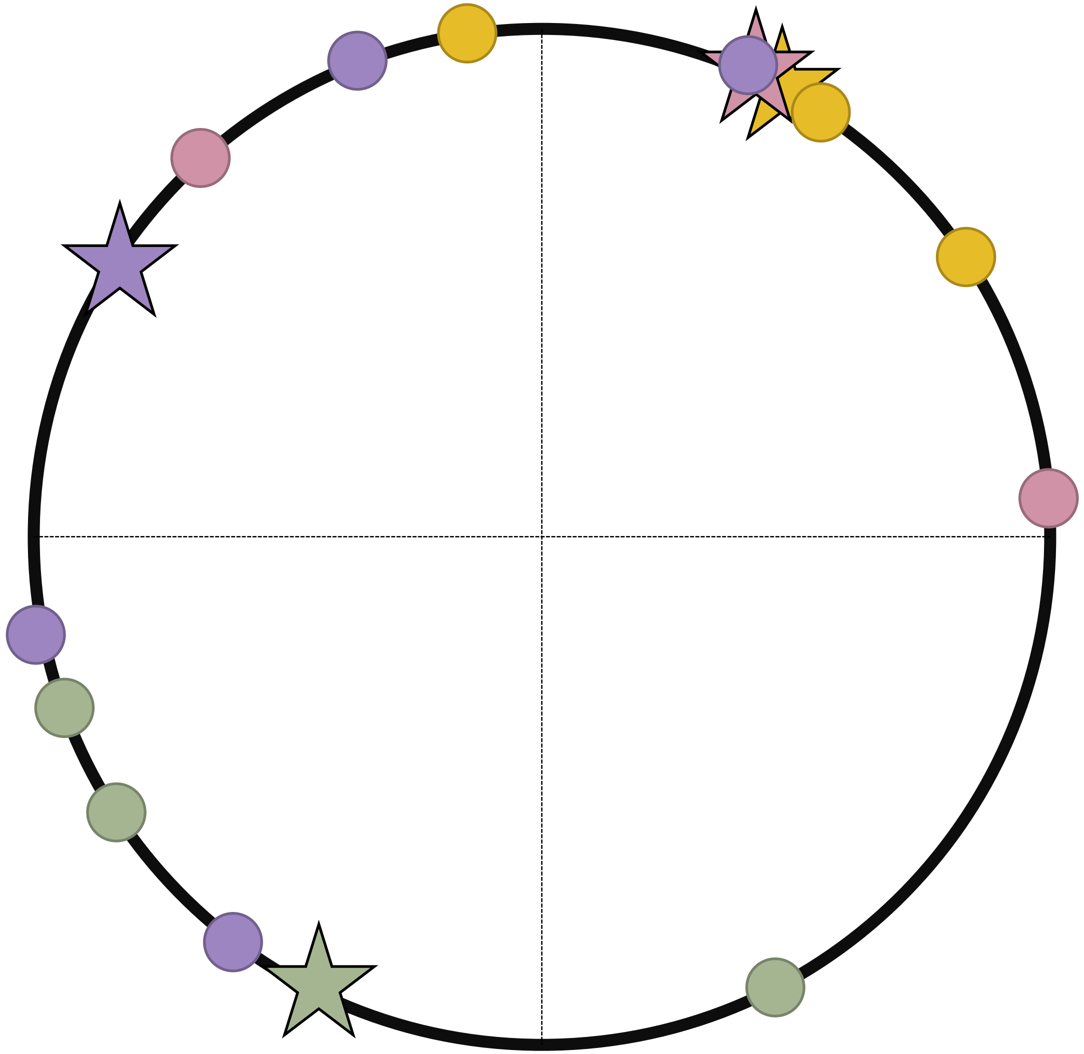
1. Choose any random K centroids
2. Forward a batch
3. Update each centroid as

$$\star' = N^{-1}\sum_{i}^{N} \bullet \quad \text{(element wise)}$$

$$\star' = \star' / \|\star'\|$$

$$\star = \alpha \star' + (\alpha - 1)\star$$

$$\star = \star / \|\star\|$$



# NCL: FINDING CENTROIDS (4)

1. Choose any random K centroids
2. Forward a batch
3. Update each centroid as

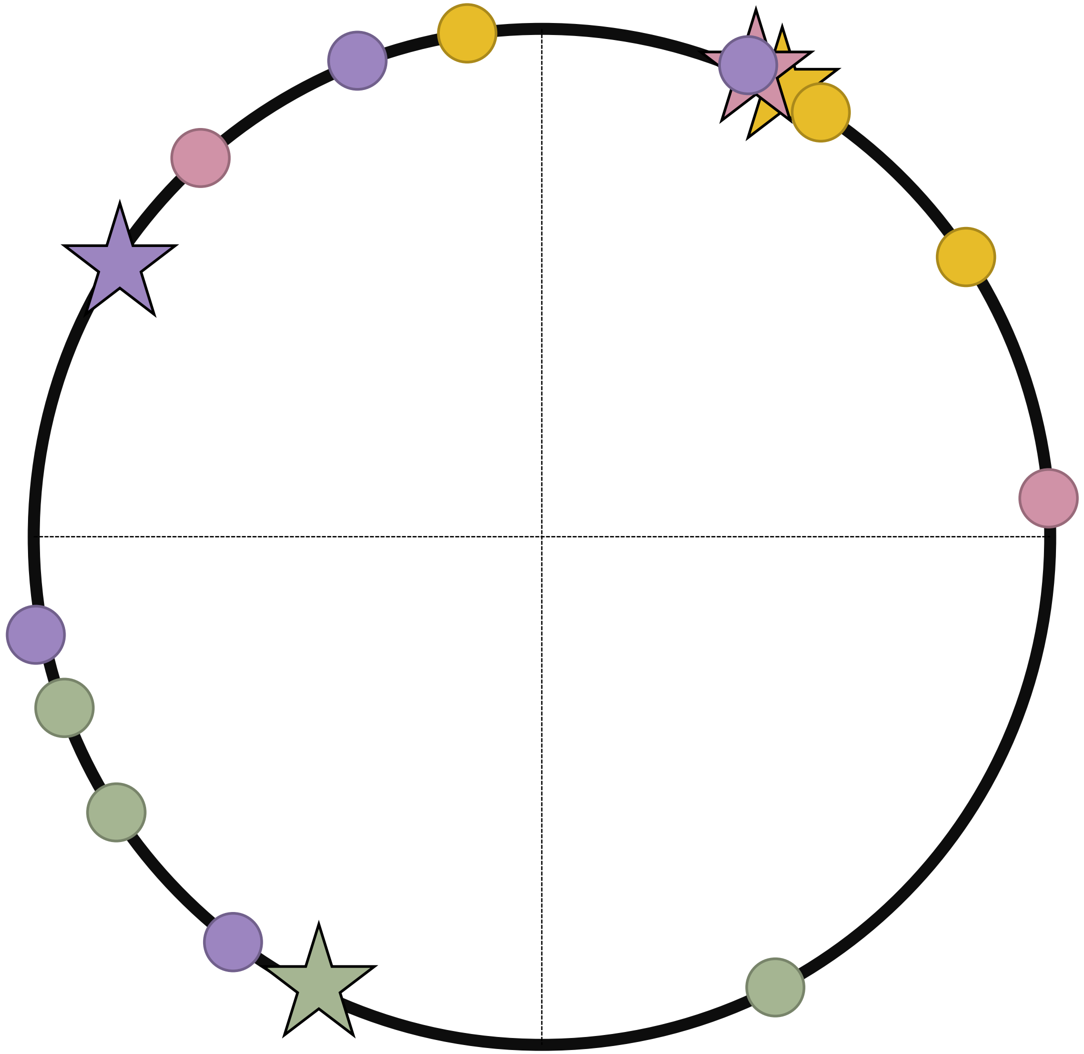
$$\star' = N^{-1} \sum_{i}^{N} \bullet \quad \text{(element wise)}$$

$$\star' = \star' / \|\star'\|$$

$$\star = \alpha \star' + (\alpha - 1)\star$$

$$\star = \star / \|\star\|$$

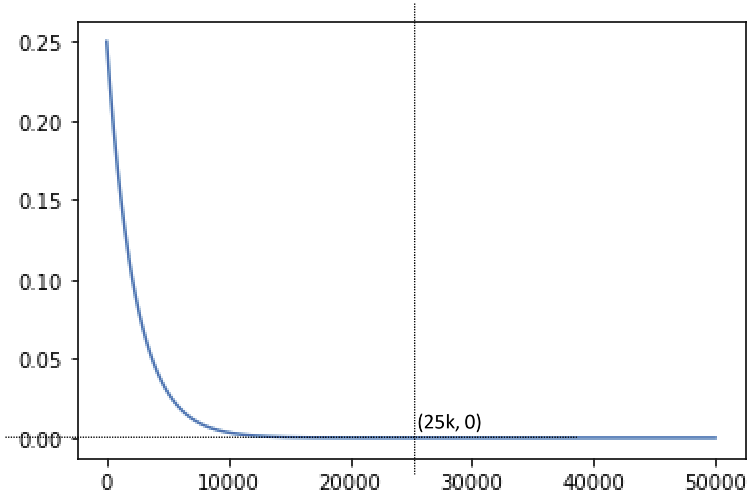
4. Backward and re-iterate from 2



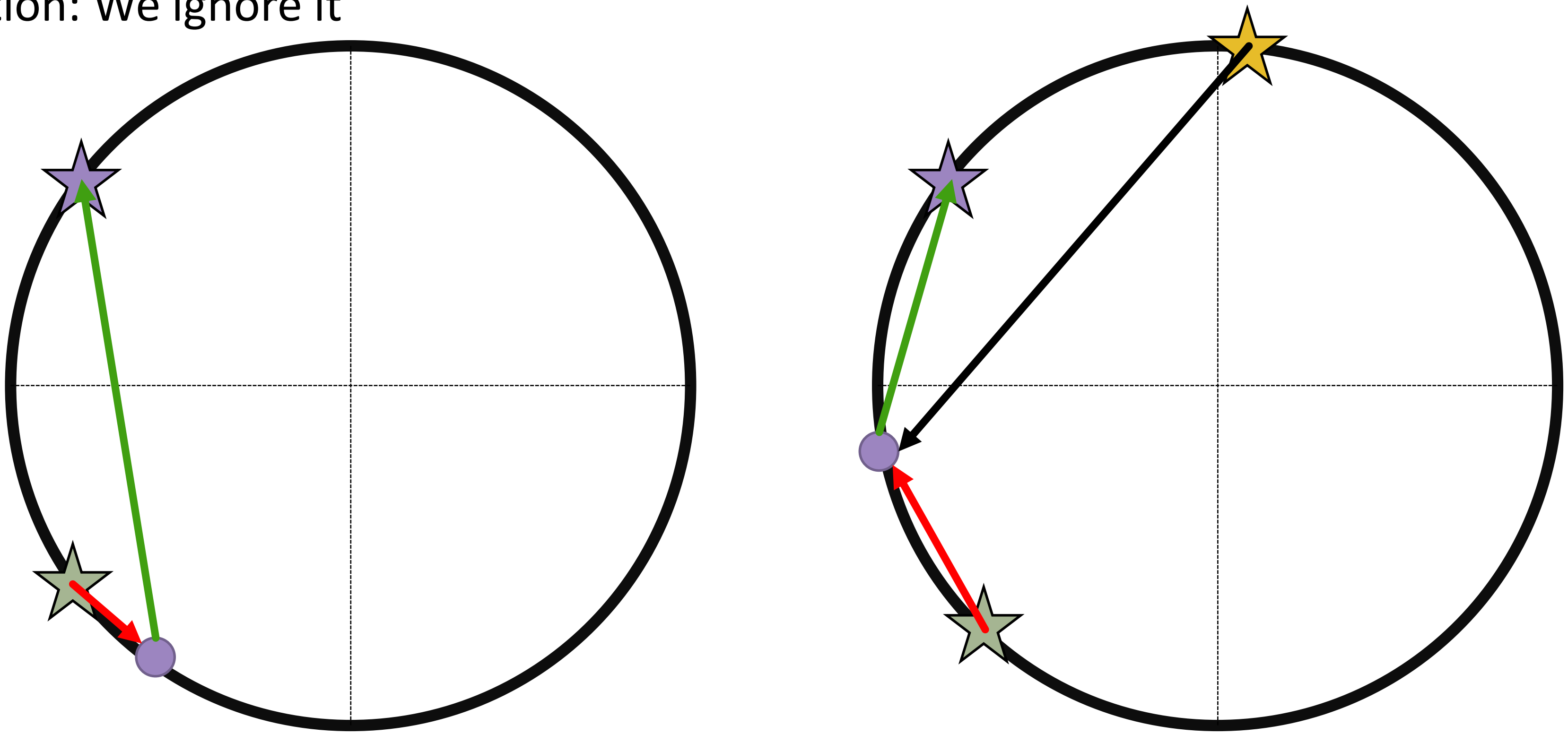
# NCL: FINDING CENTROIDS ($\alpha$)

- Prefer fast modifications at first
- Decay to 0 but allow modifications

$$\alpha = 0.25 \cdot 0.65^{iter/1000}$$



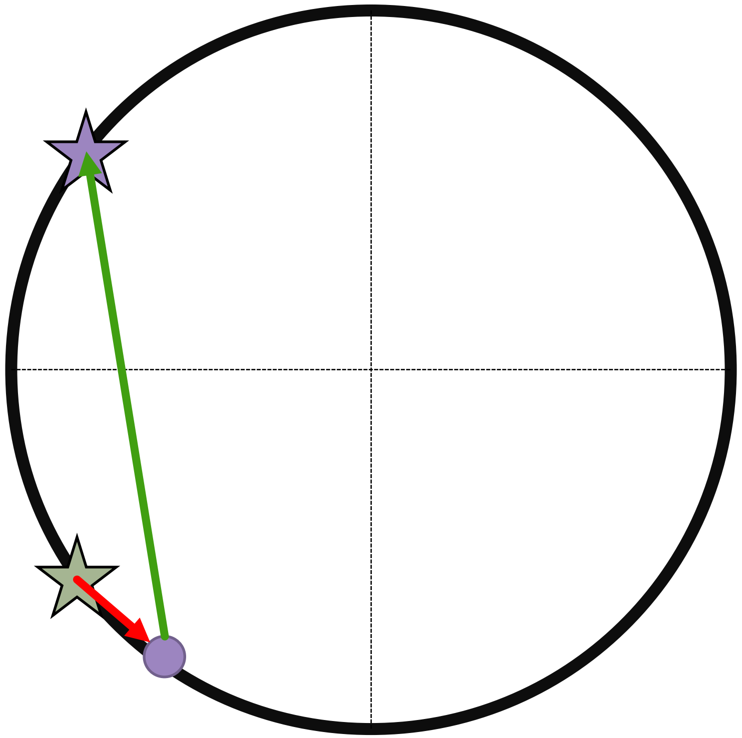
- Consider the following cases
  - Problem: Opposing terms in the loss
  - Solution: We ignore it



# NCL: REASON 1

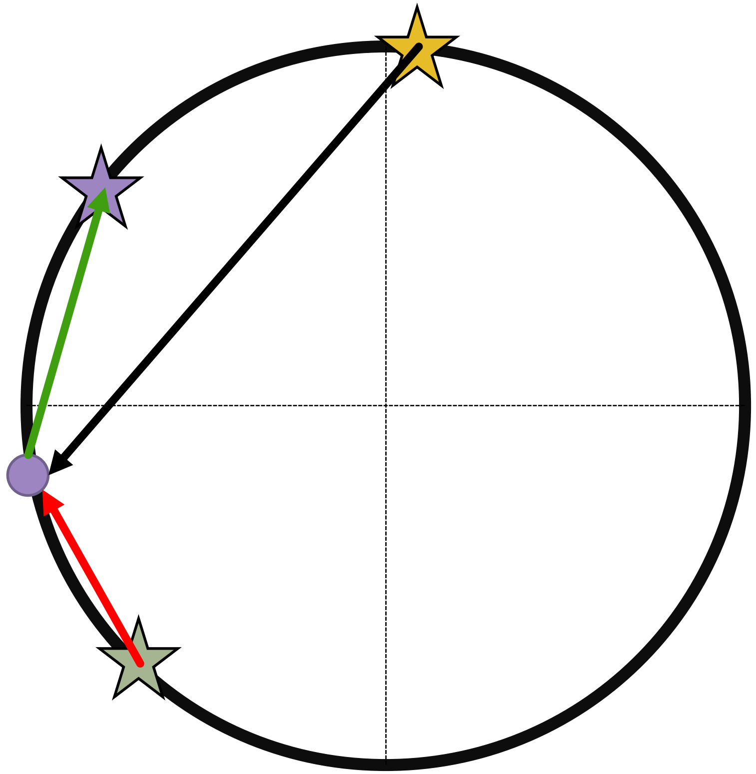
$d(A, P) > d(A, N)$

It will move to its centroid



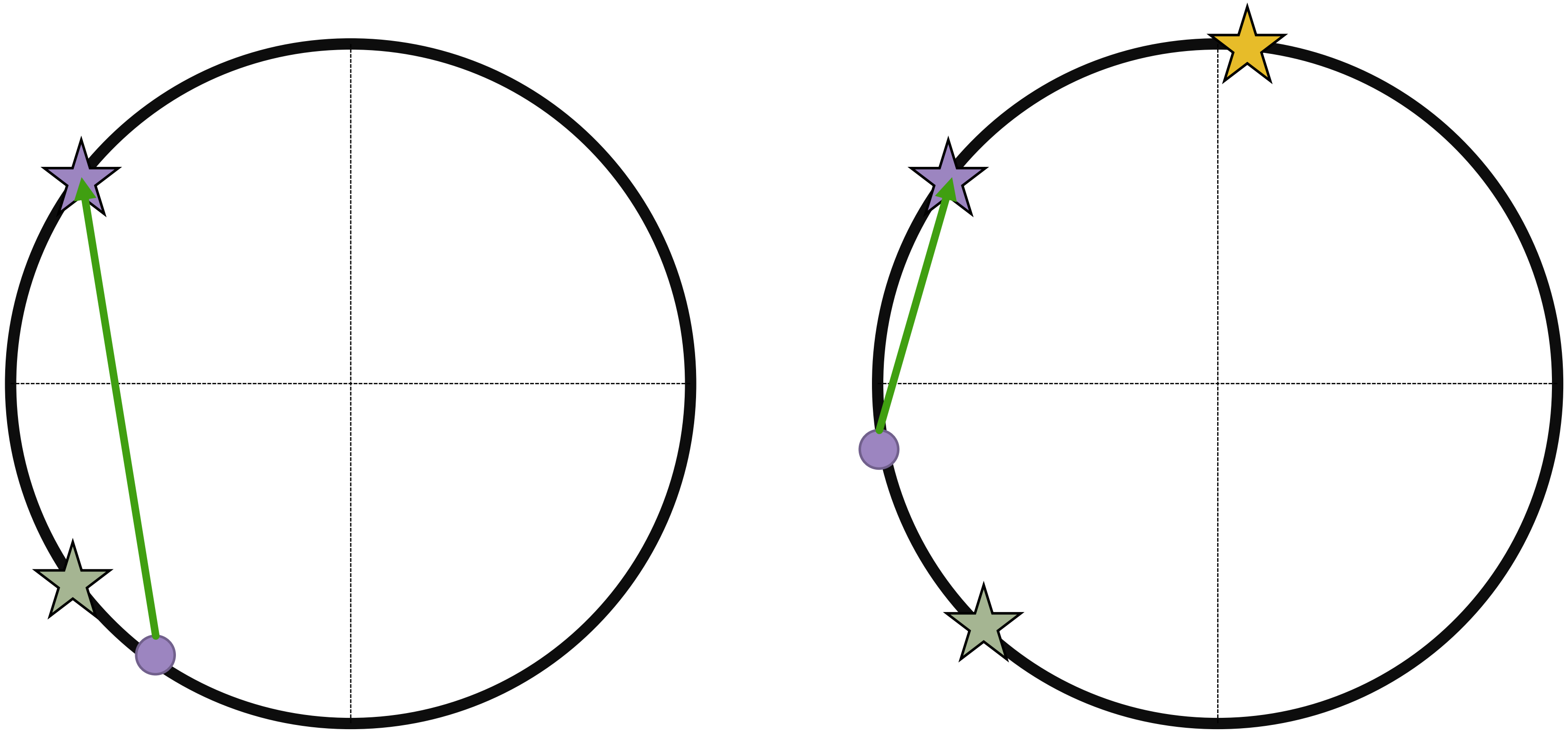
$d(A, P),\ d(A, N)$ "same direction"

N-d might be a problem

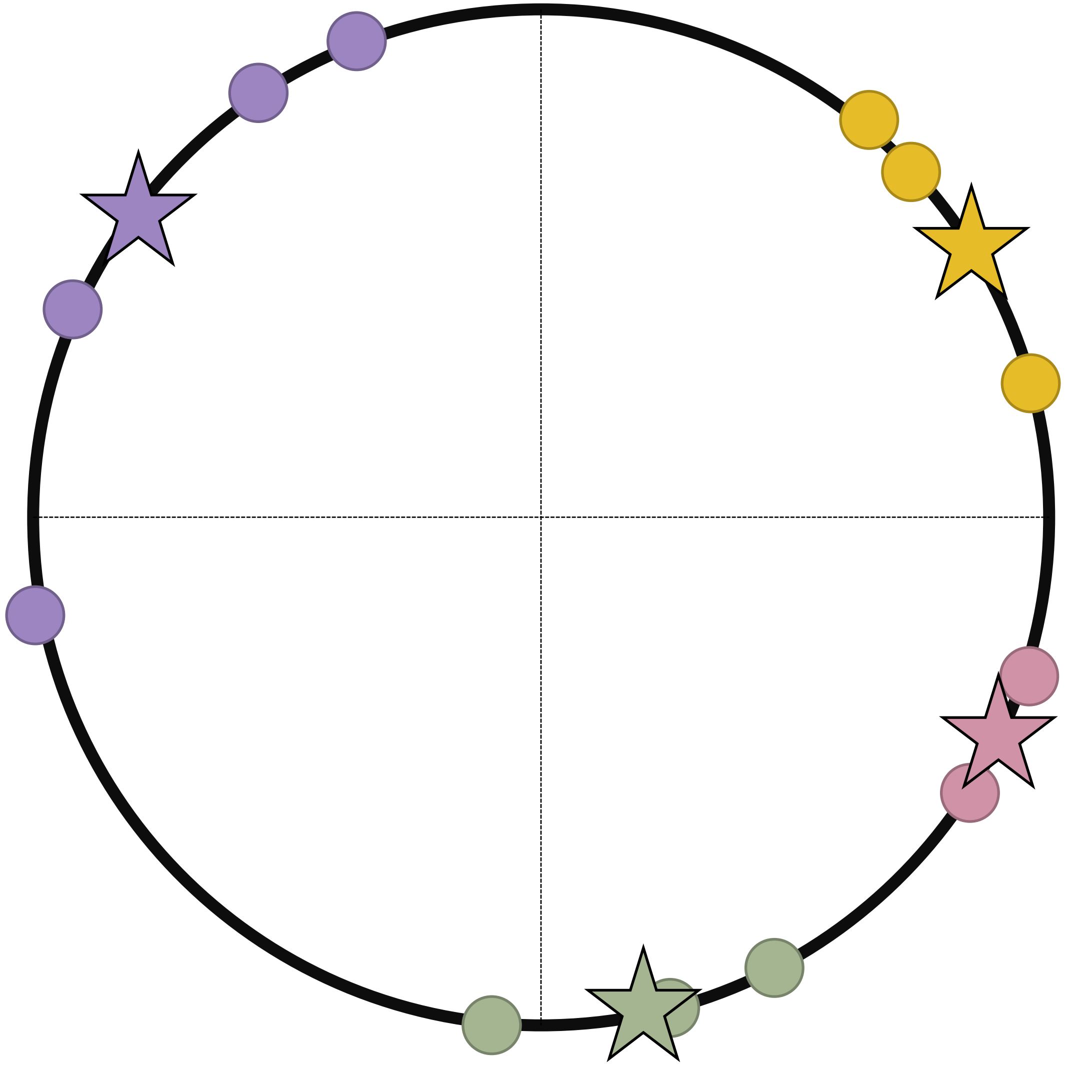


# NCL: REASON 2

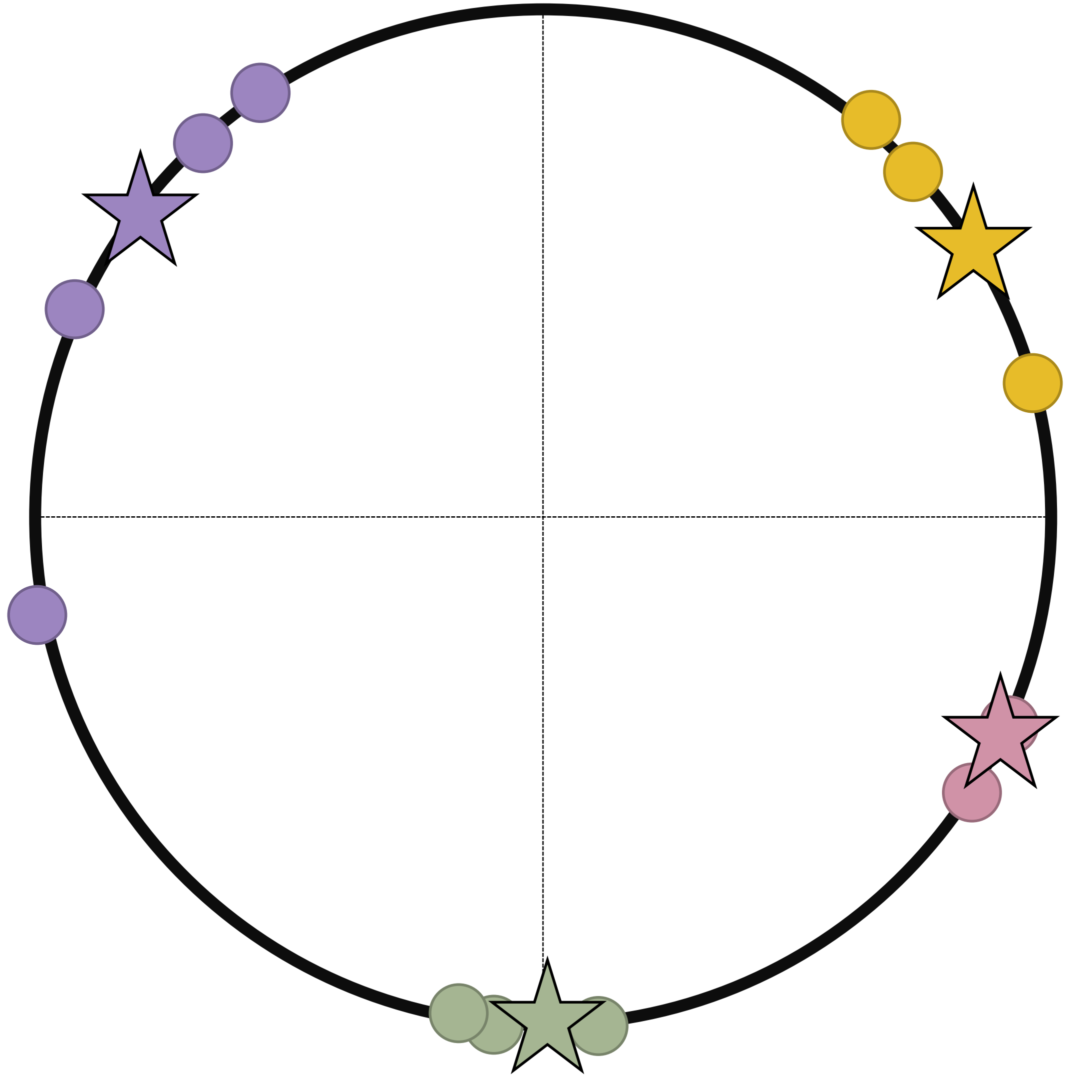
- I lied, the loss is $d(\star,\bullet) - \underbrace{(\boldsymbol{\alpha} > \mathbf{0})}_{\text{When } \alpha \text{ is 0, we don't have a negative term}} \cdot \left( \mathrm{d}(\star,\bullet) + \mathrm{K}^{-1} \sum_{i=0}^{K} d(\star \neq \star,\bullet) \right)$



# NCL: STUDY CASE



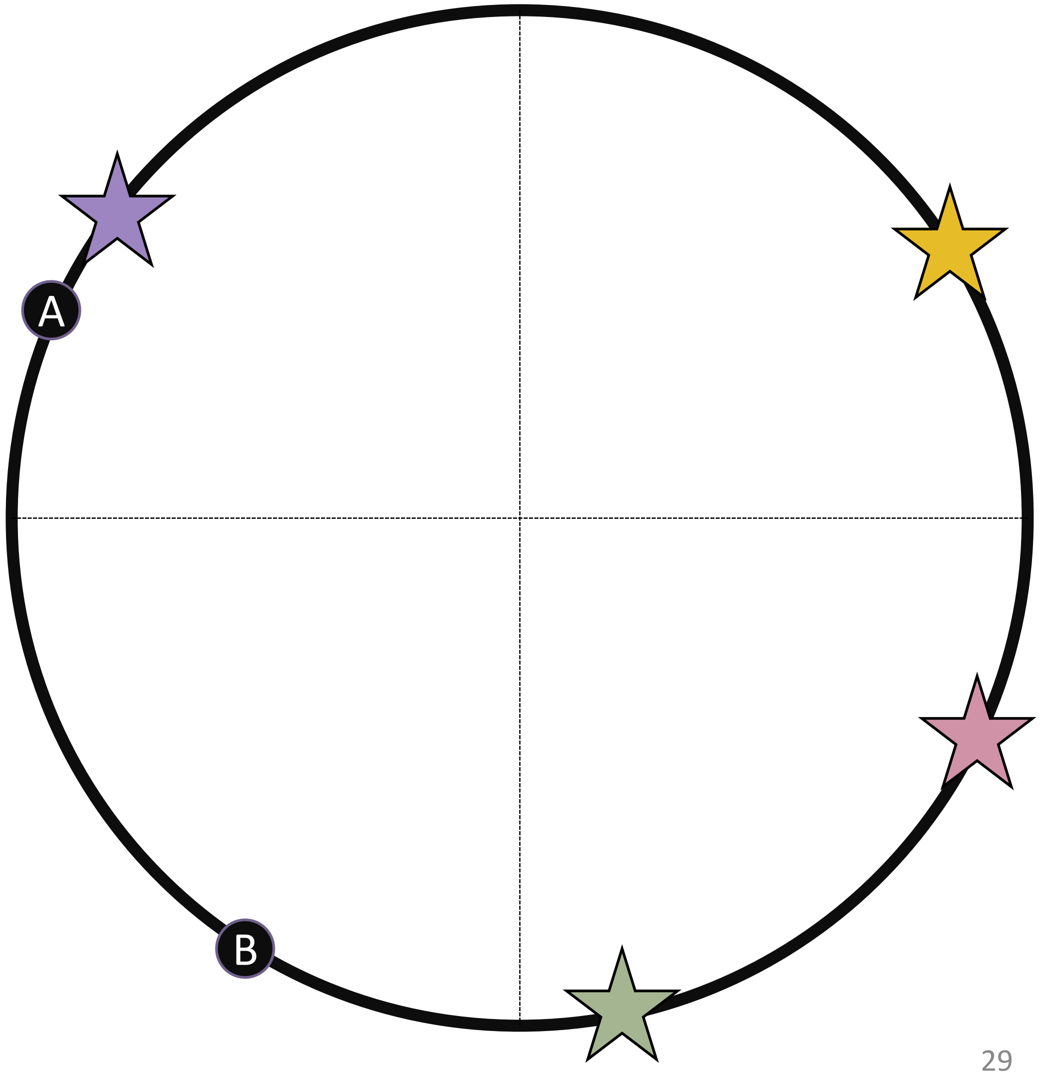
# NCL: STUDY CASE



# CONFIDENCE ESTIMATE

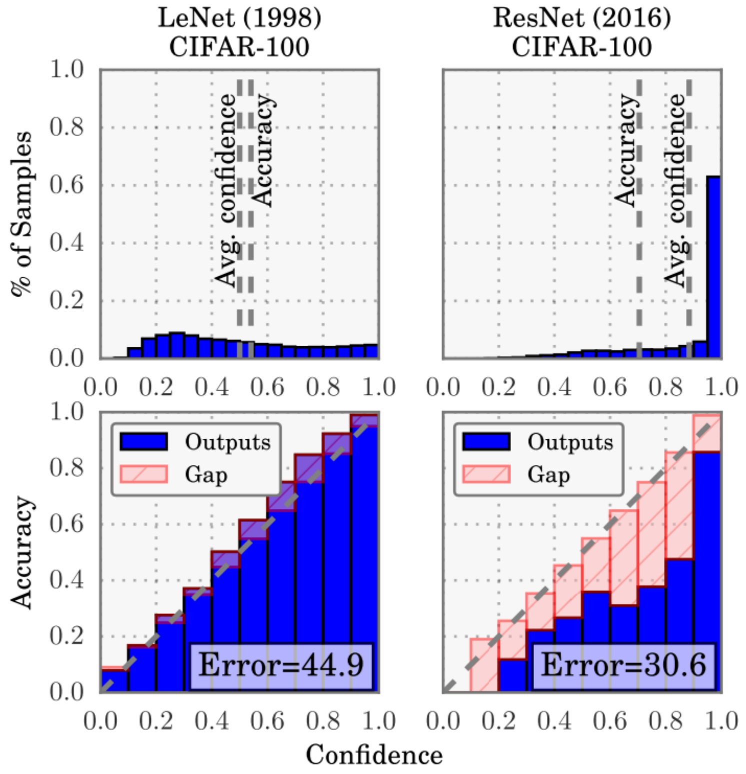
# CE: BACK TO NCL

- Both (A) and (B) are eval/test
- How good is their embedding?
  1. Look at classification prob.
  2. Check proximity to nearest centroid



# CE: $P(y|x)$

1. Look at classification probability
    - NNs output probabilities are not well calibrated [1]
    - NNs tend to be overconfident under dataset shifts and out of distribution samples [2]







[1] GUO, Chuan, et al. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.
[2] OVADIA, Yaniv, et al. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. On *Advances in Neural Information Processing Systems*. 2019. p. 13991-14002.

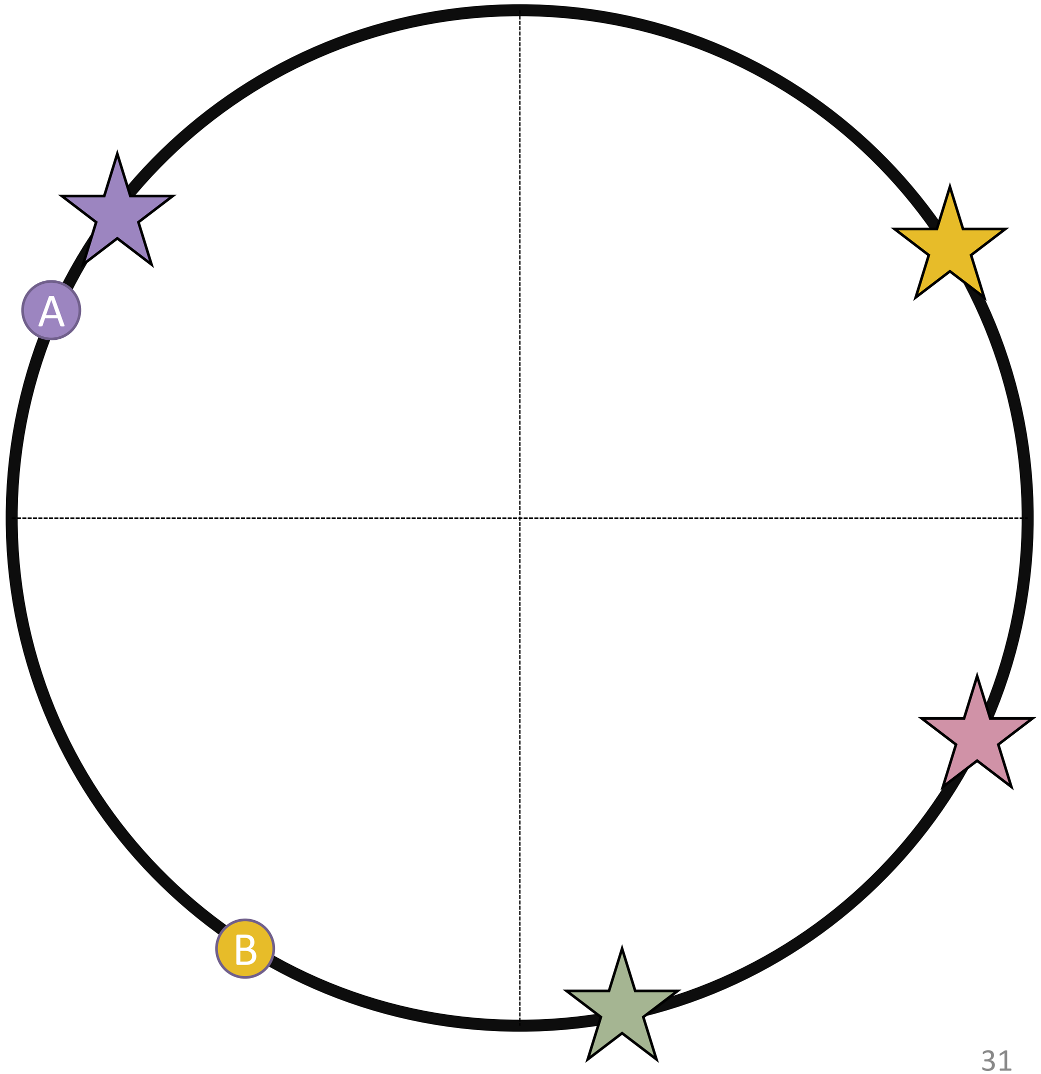
# CE: PROXIMITY

2. Check proximity to nearest centroid
   - Naïve!

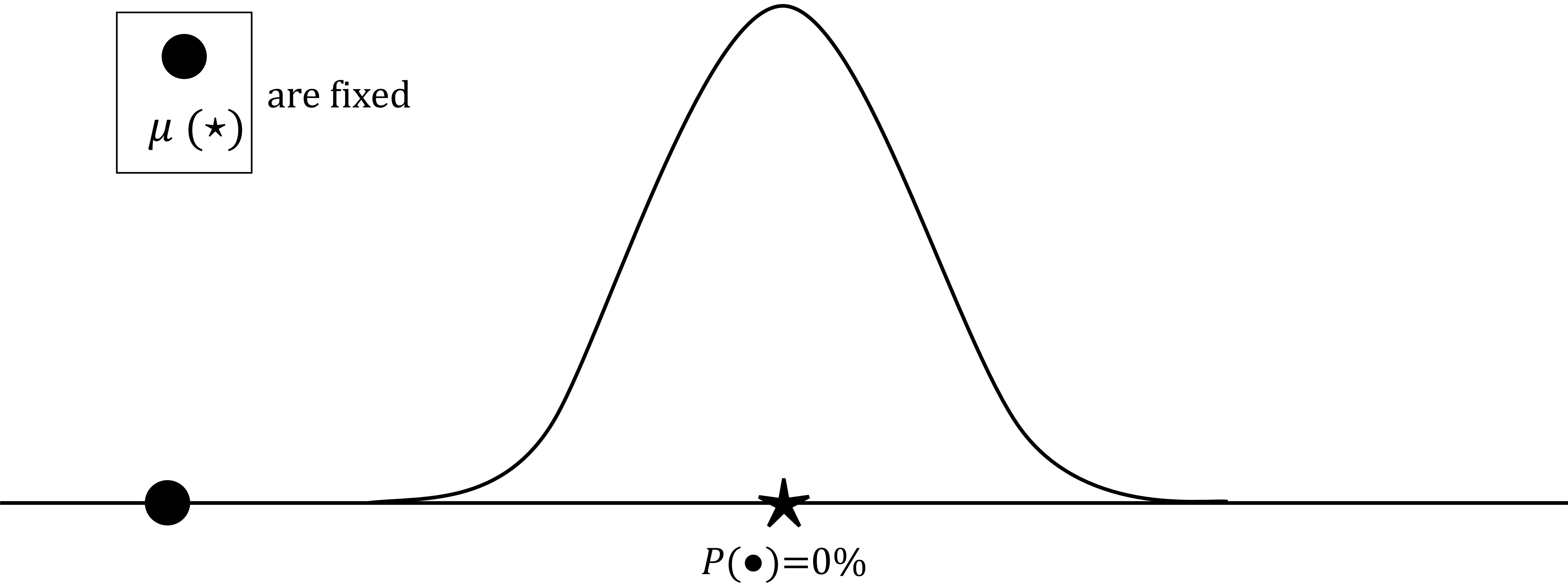
$$C(A) = 90\%$$

$$C(B) = 60\%$$

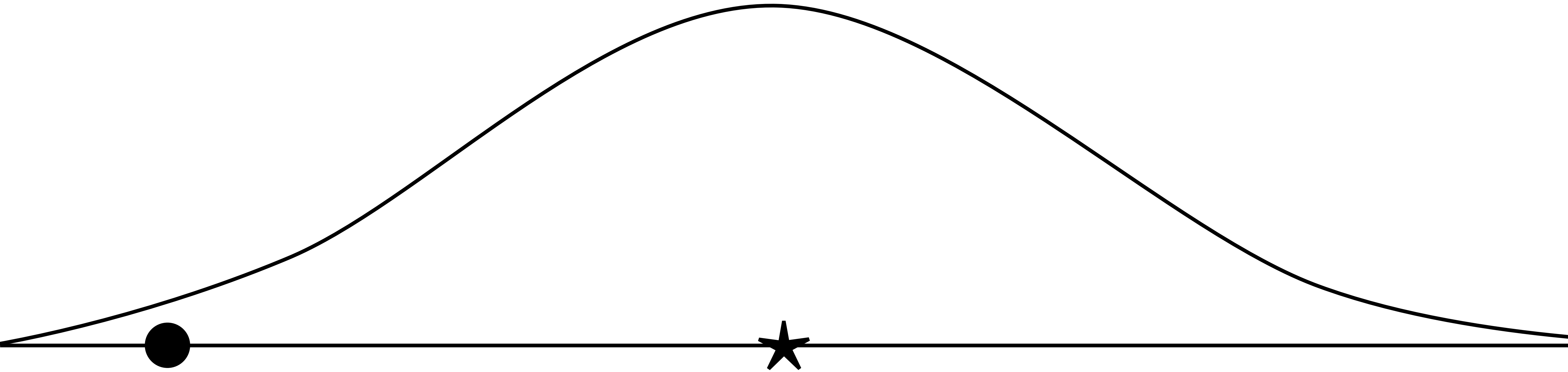
   - It only works if the embedding is perfect. At which point, why would you need confidence?



# CE: $N(\mu, \sigma)$ TO THE RESCUE

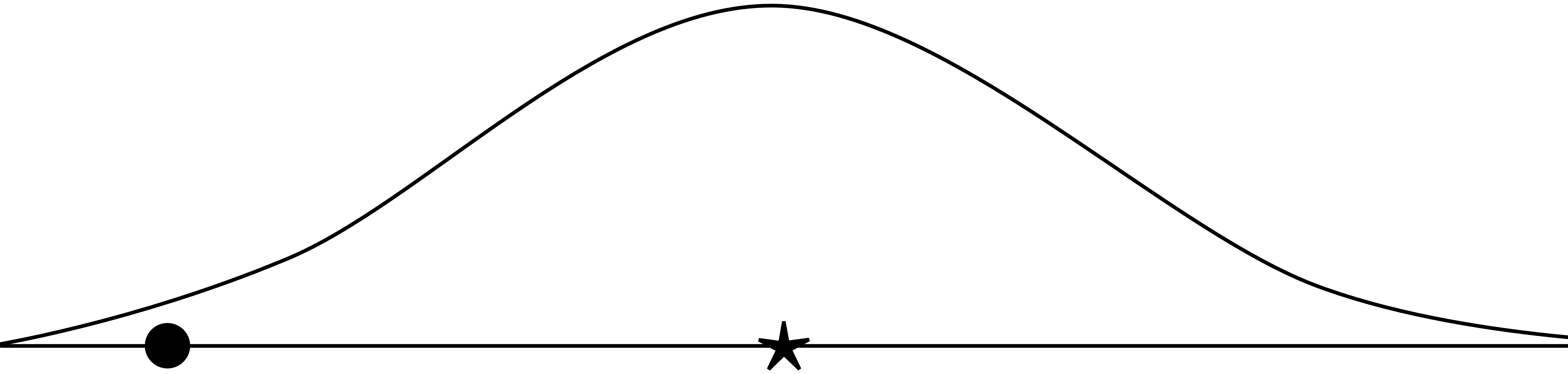


# CE: 1D SAMPLE CASE



$P(\bullet)$=5%

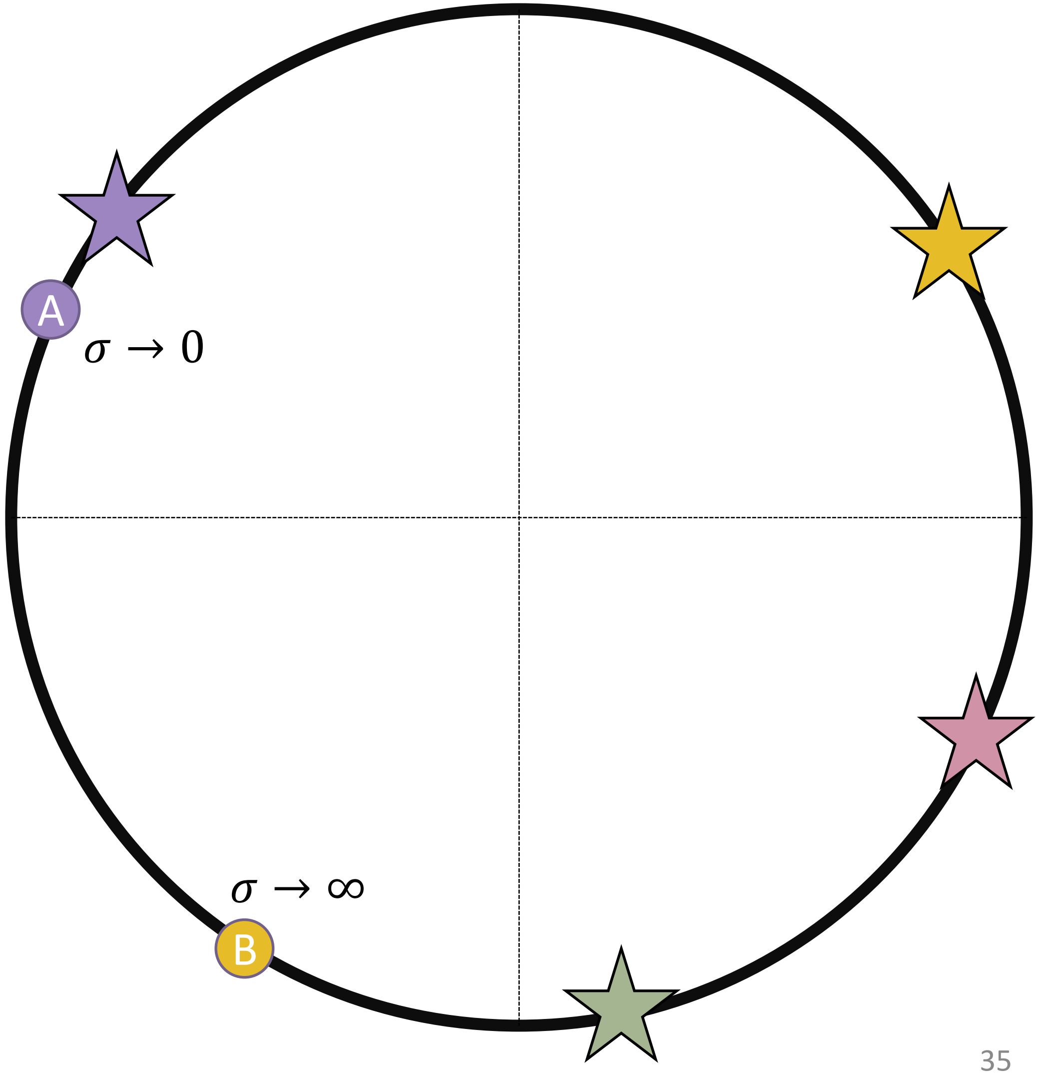
# CE: 1D SAMPLE CASE



$\sigma \uparrow \Rightarrow P(\bullet) \uparrow$

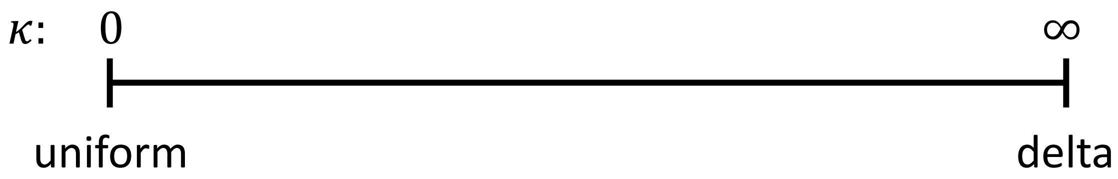
# CE: WHAT THEN?

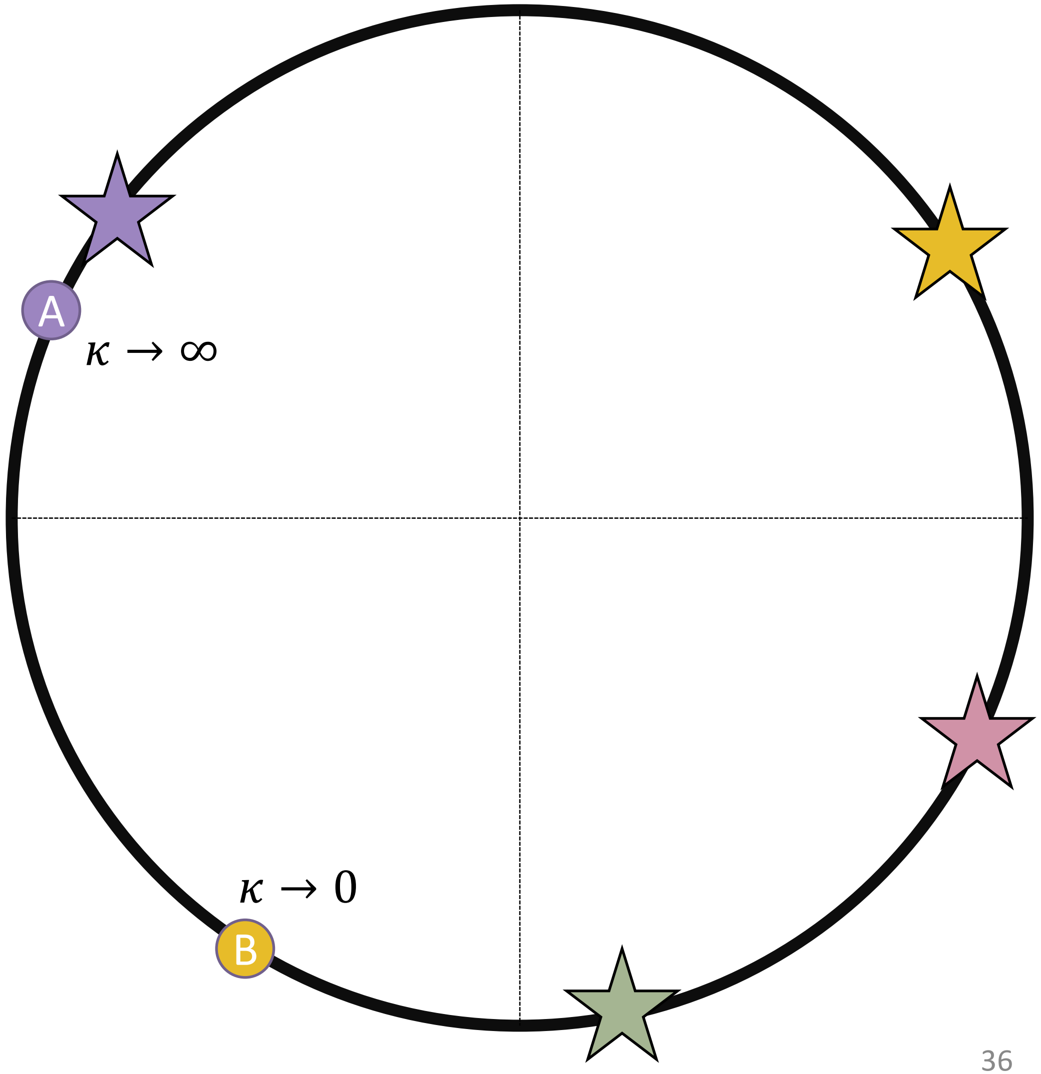
- During training
  - Place a Normal with diagonal covariance at the sample's centroid
  - Optimize for the $\sigma$ that gives the highest probability for each sample
    - Negative Log Likelihood (NLL)
- During inference
  - Use $\sigma$ as a threshold (i.e. valid if > x)



# CE: CAN WE DO BETTER?

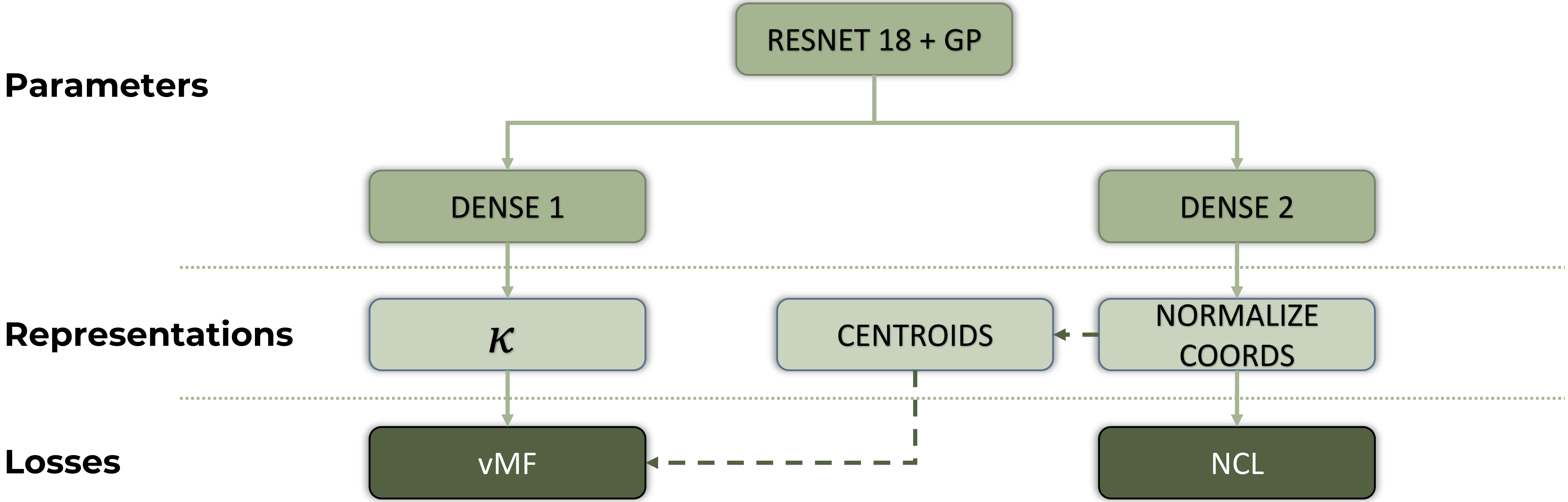
- Normal distributions work over the whole space
- Switch to a distribution specialized to spheres
  - Von Mises-Fisher
    - $\mu \in \mathbb{R}^p$ mean direction
    - $\kappa \in \mathbb{R}$ concentration



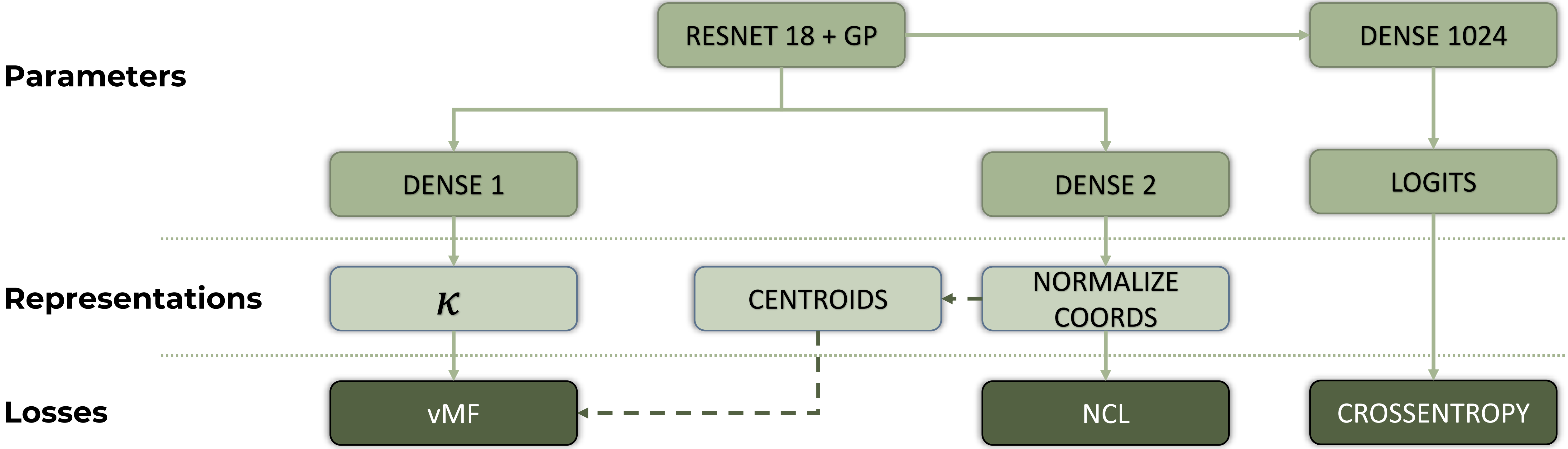
# ARCHITECTURE

# ARCHITECTURE: v1



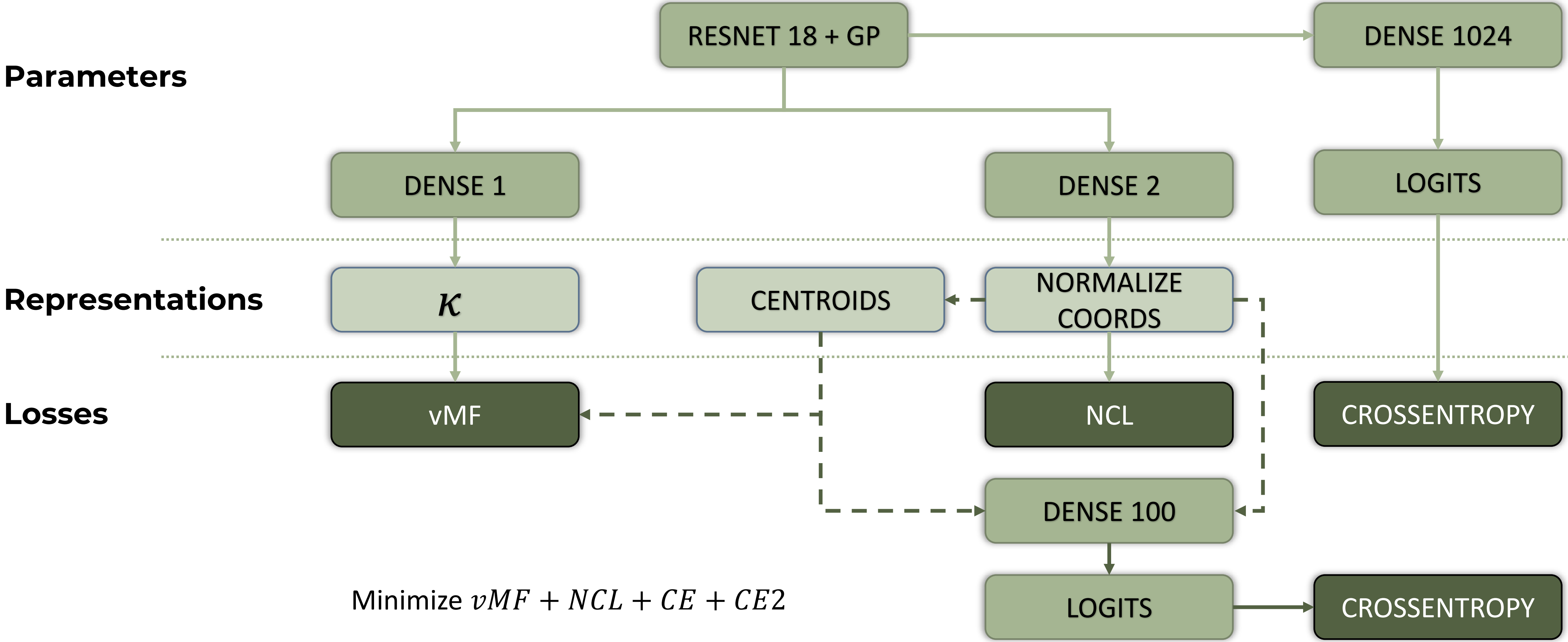

Minimize $vMF + NCL$

# ARCHITECTURE: v2




Minimize $vMF + NCL + CE$

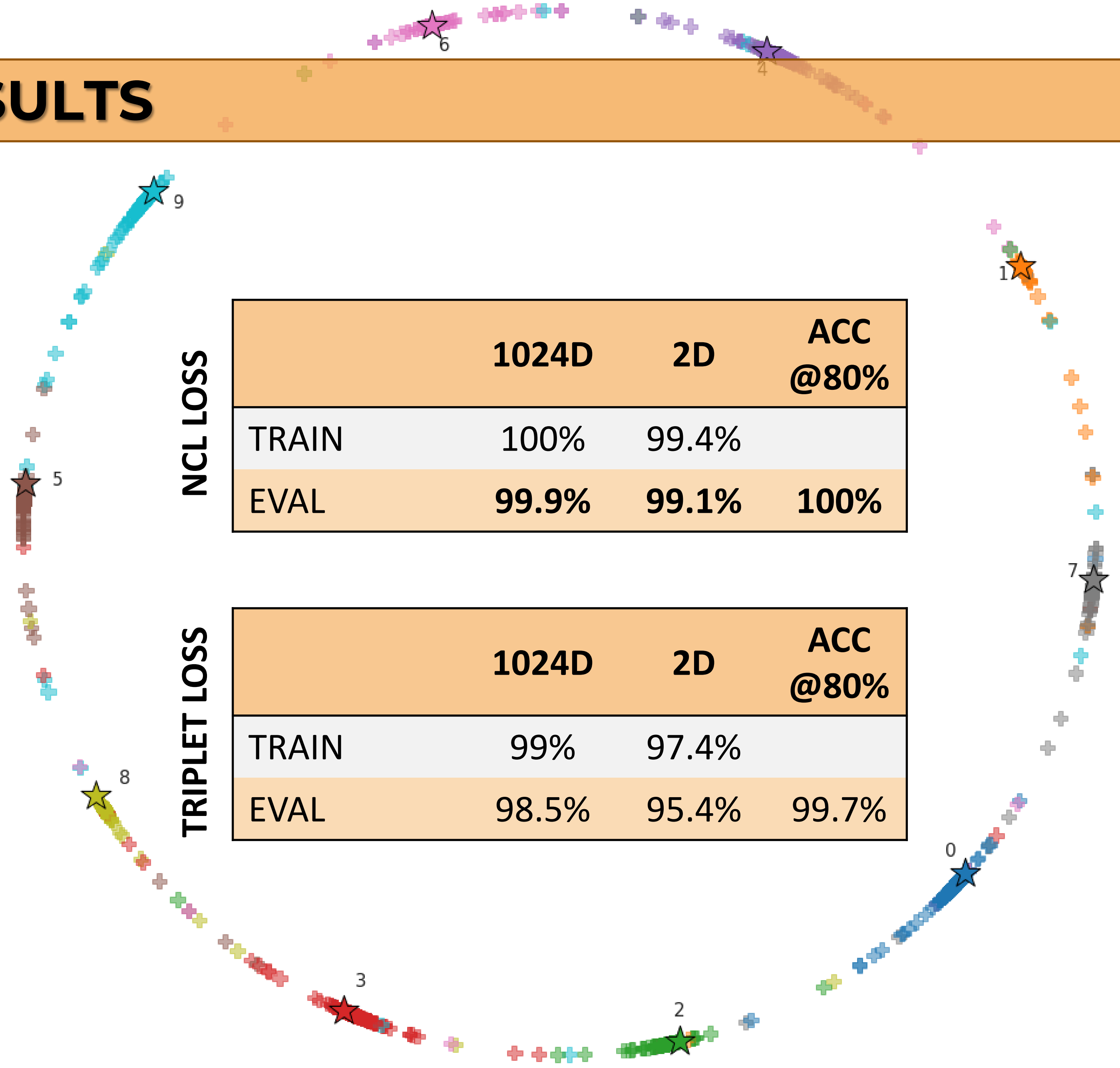
# ARCHITECTURE: EMBEDDING CLS




Minimize $vMF + NCL + CE + CE2$

# RESULTS (MNIST)

# MNIST RESULTS



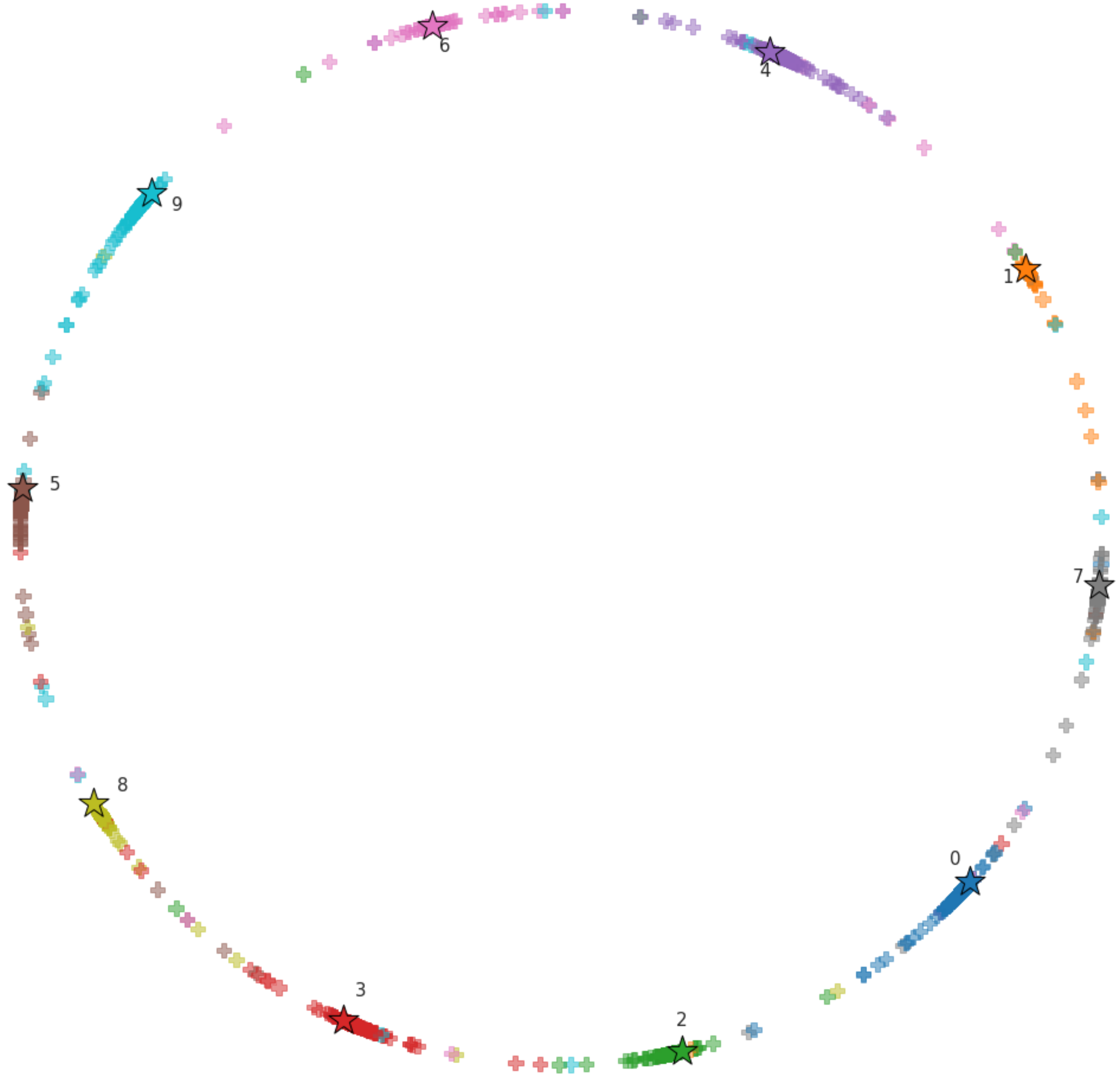

**NCL LOSS**

| | 1024D | 2D | ACC @80% |
|---|---|---|---|
| TRAIN | 100% | 99.4% | |
| EVAL | **99.9%** | **99.1%** | **100%** |

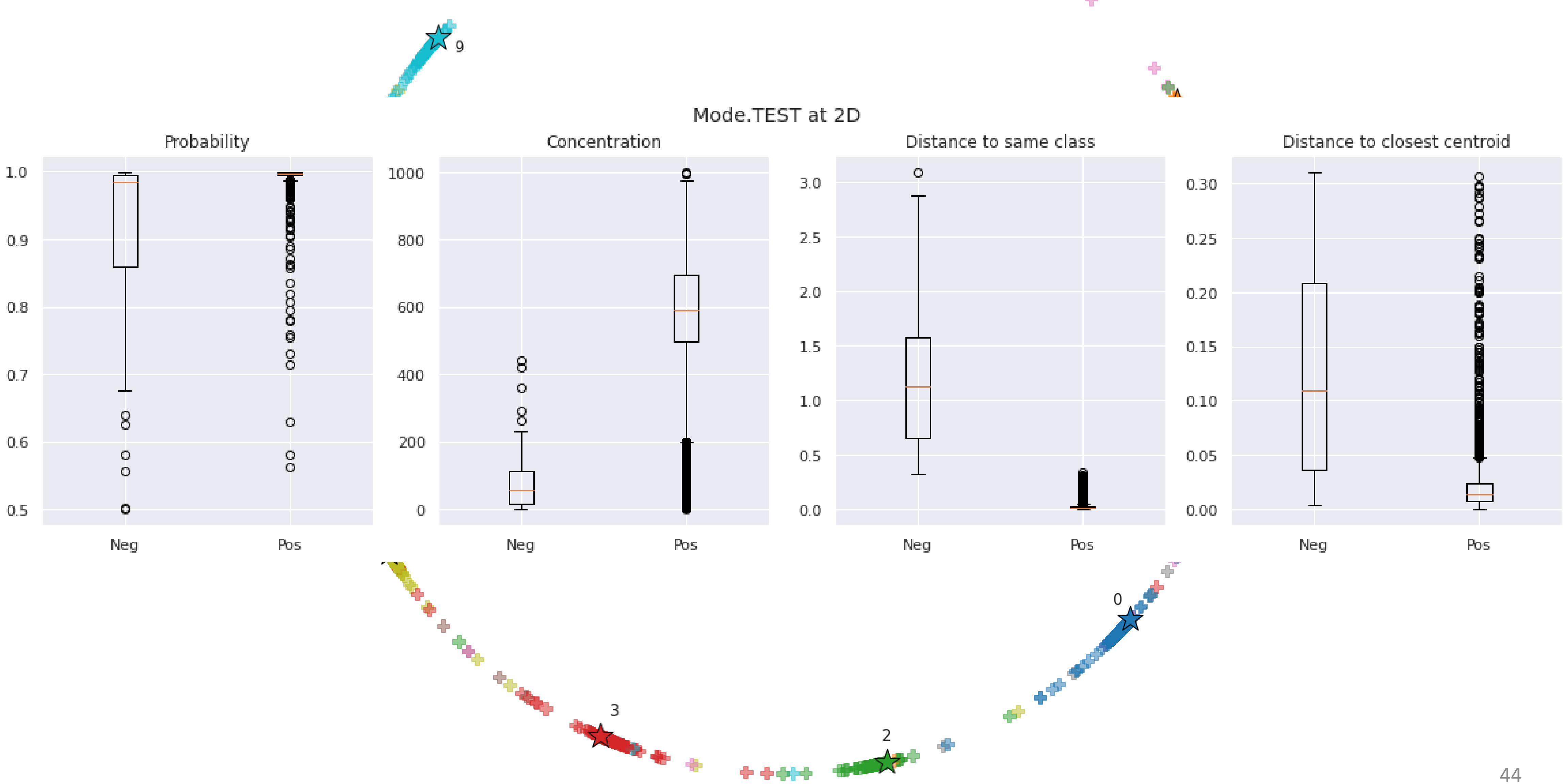
**TRIPLET LOSS**

| | 1024D | 2D | ACC @80% |
|---|---|---|---|
| TRAIN | 99% | 97.4% | |
| EVAL | 98.5% | 95.4% | 99.7% |

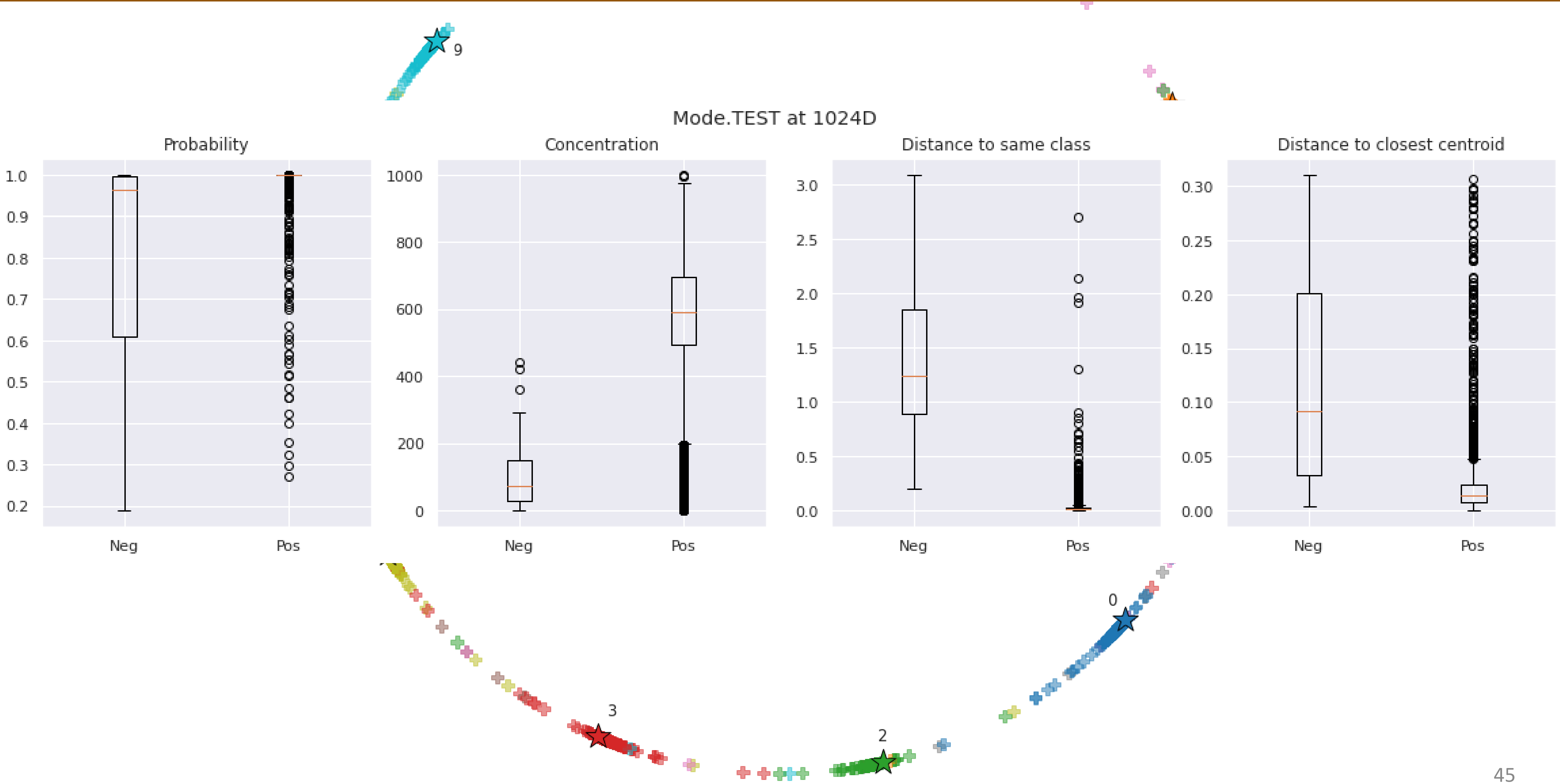
# MNIST 2D EMBEDDING



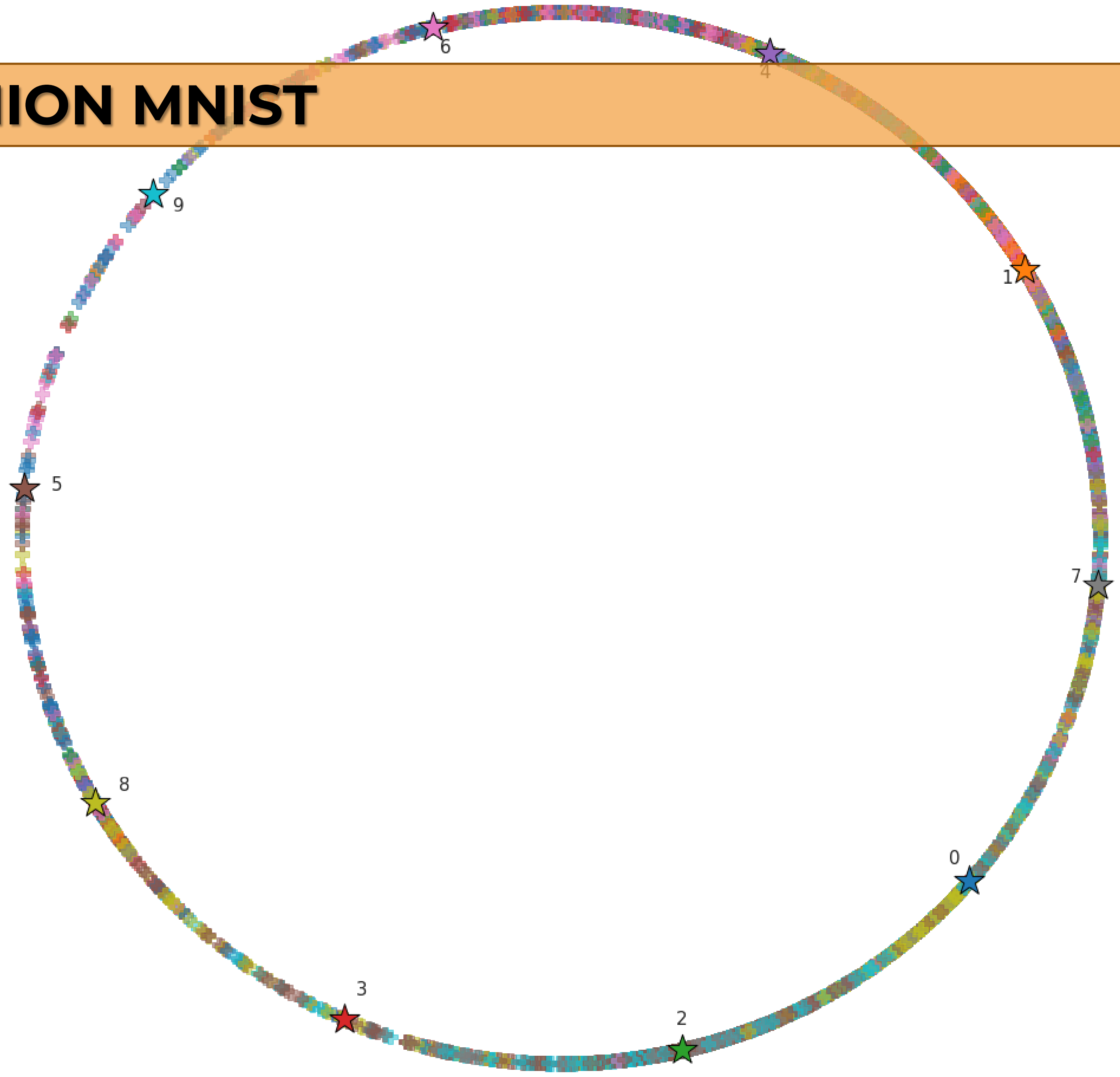
# MNIST CONF AT 2D



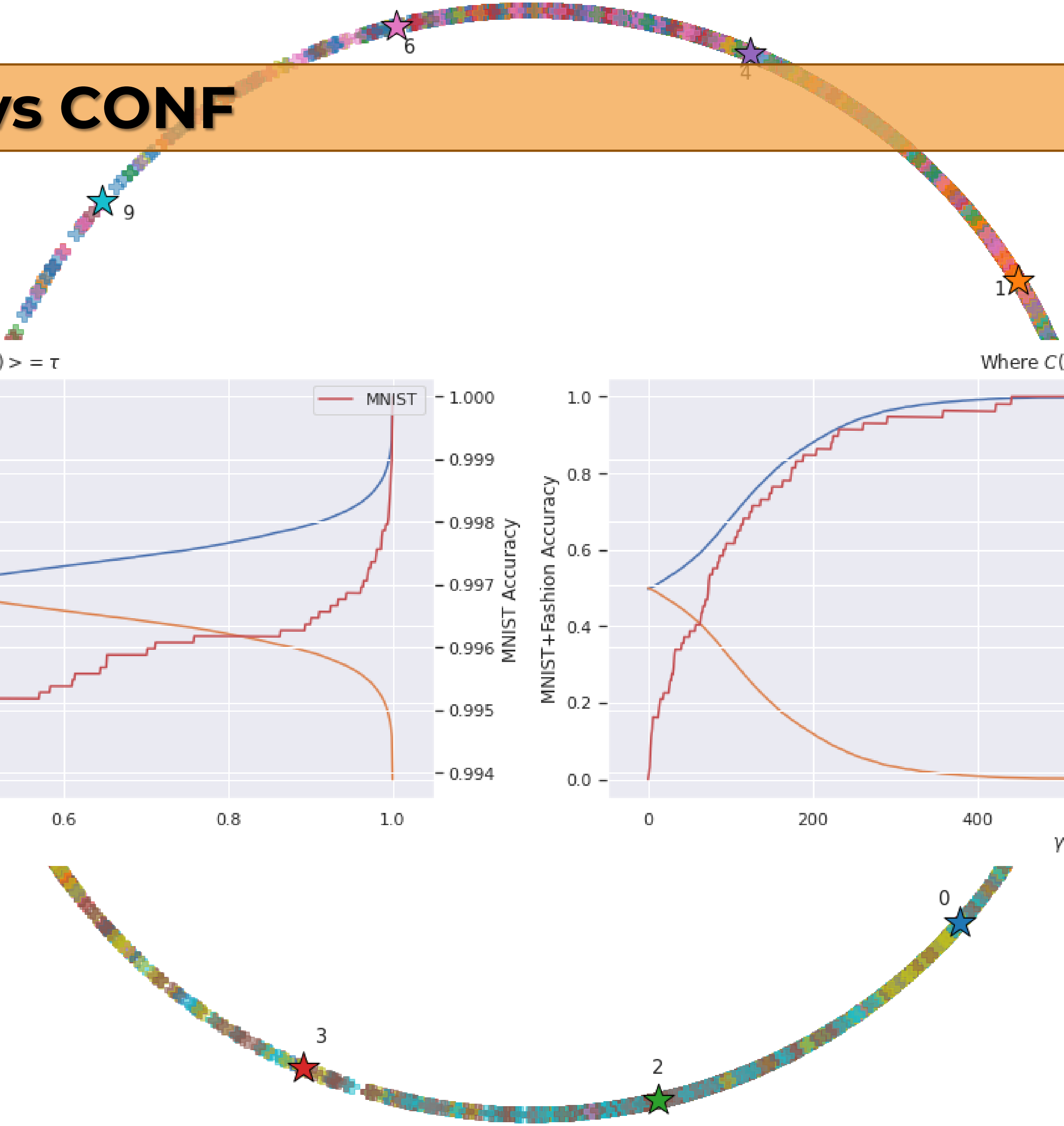
# MNIST CONF AT 1024D

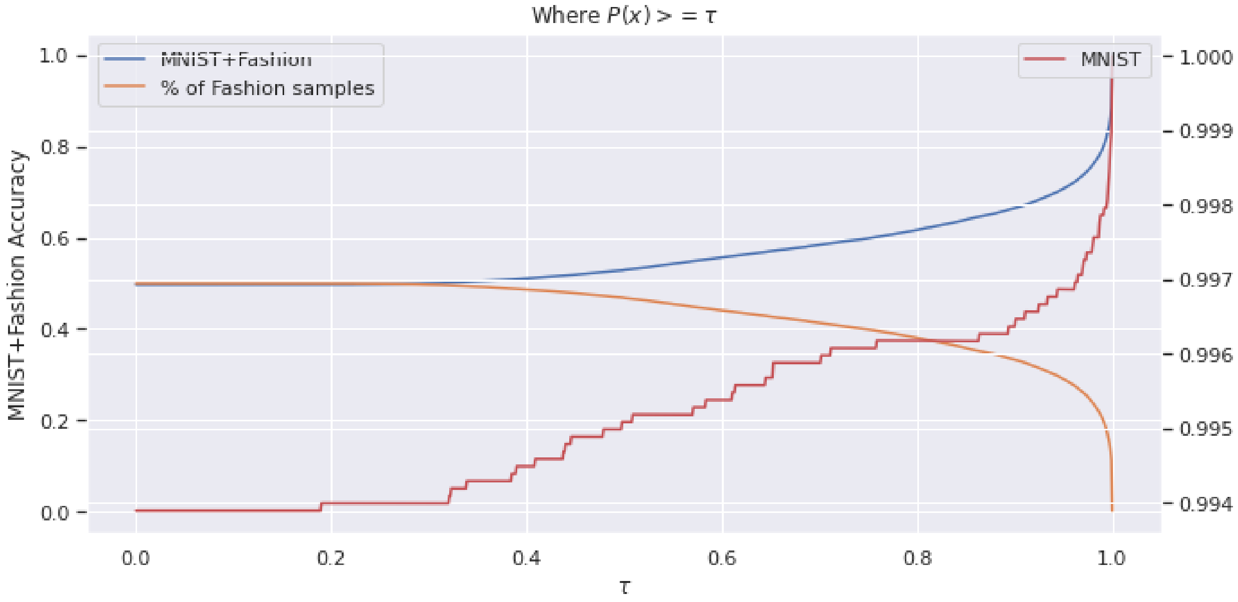


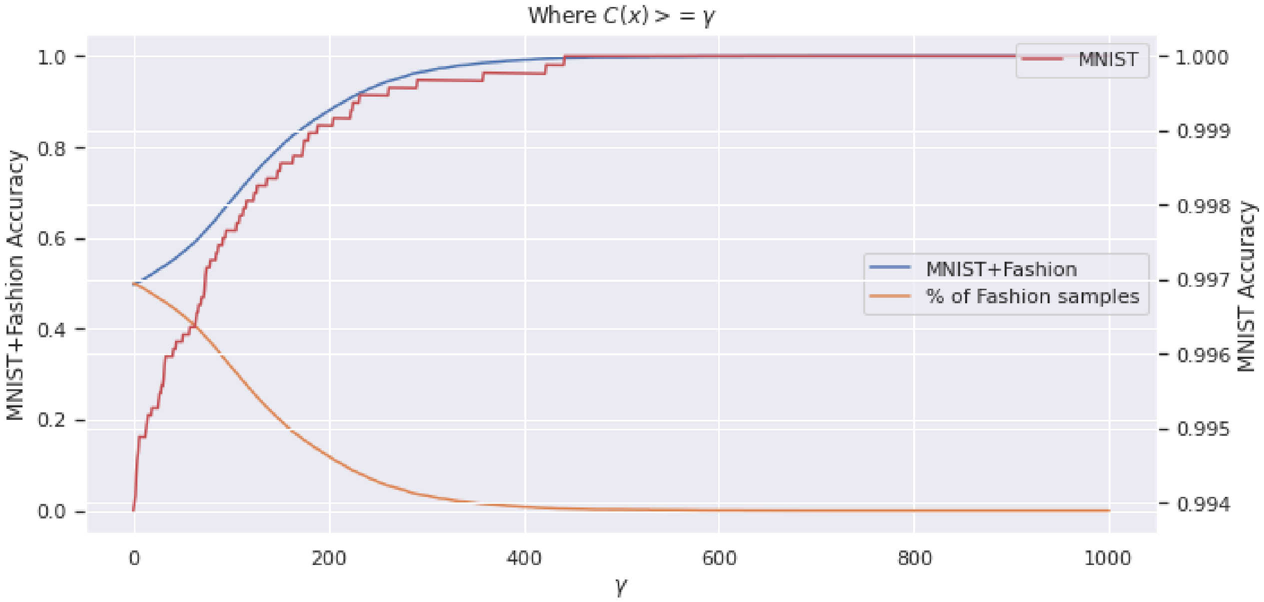
# OOD FASHION MNIST



# OOD PROB vs CONF

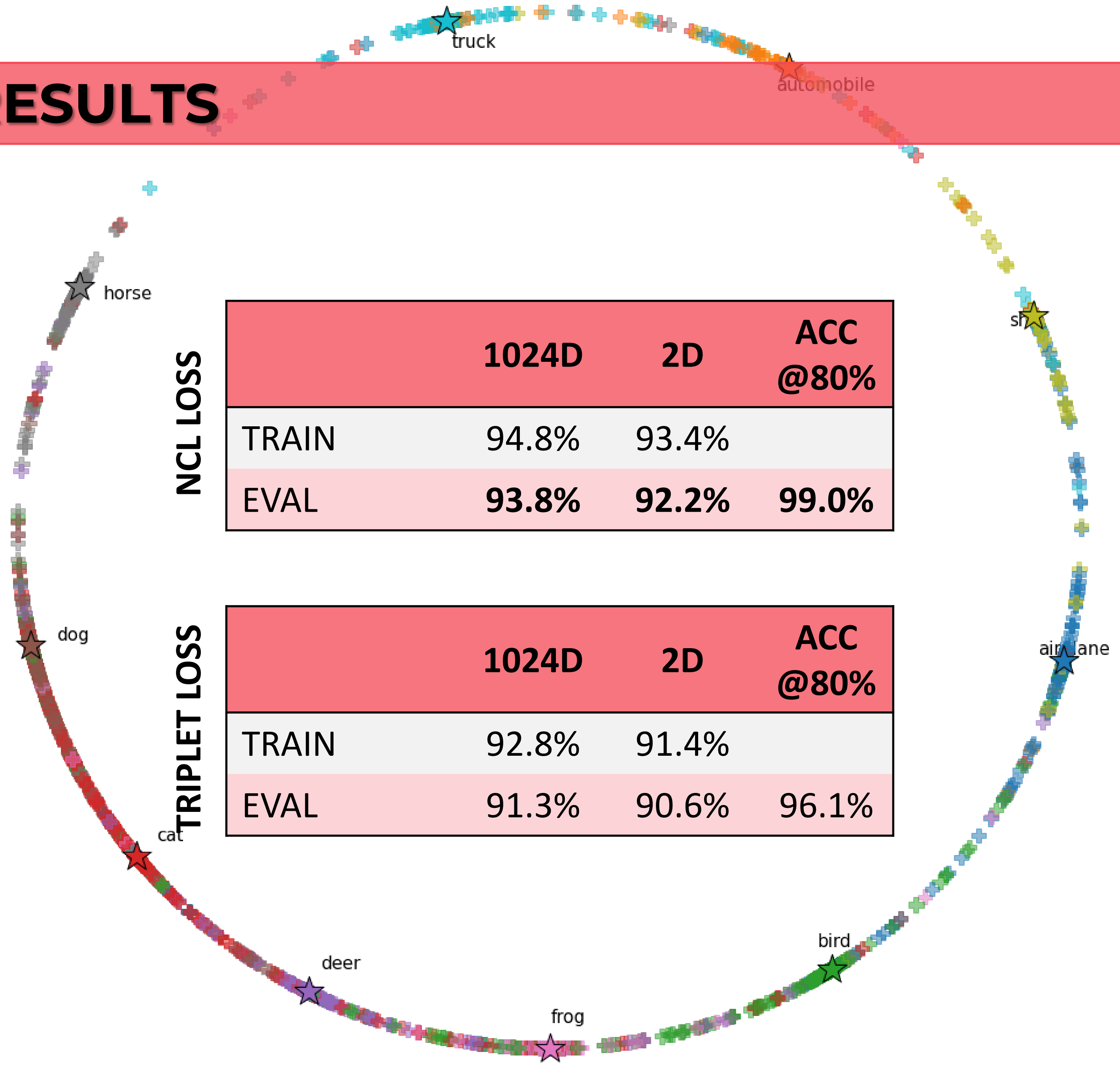


# RESULTS (CIFAR-10)

# CIFAR-10 RESULTS



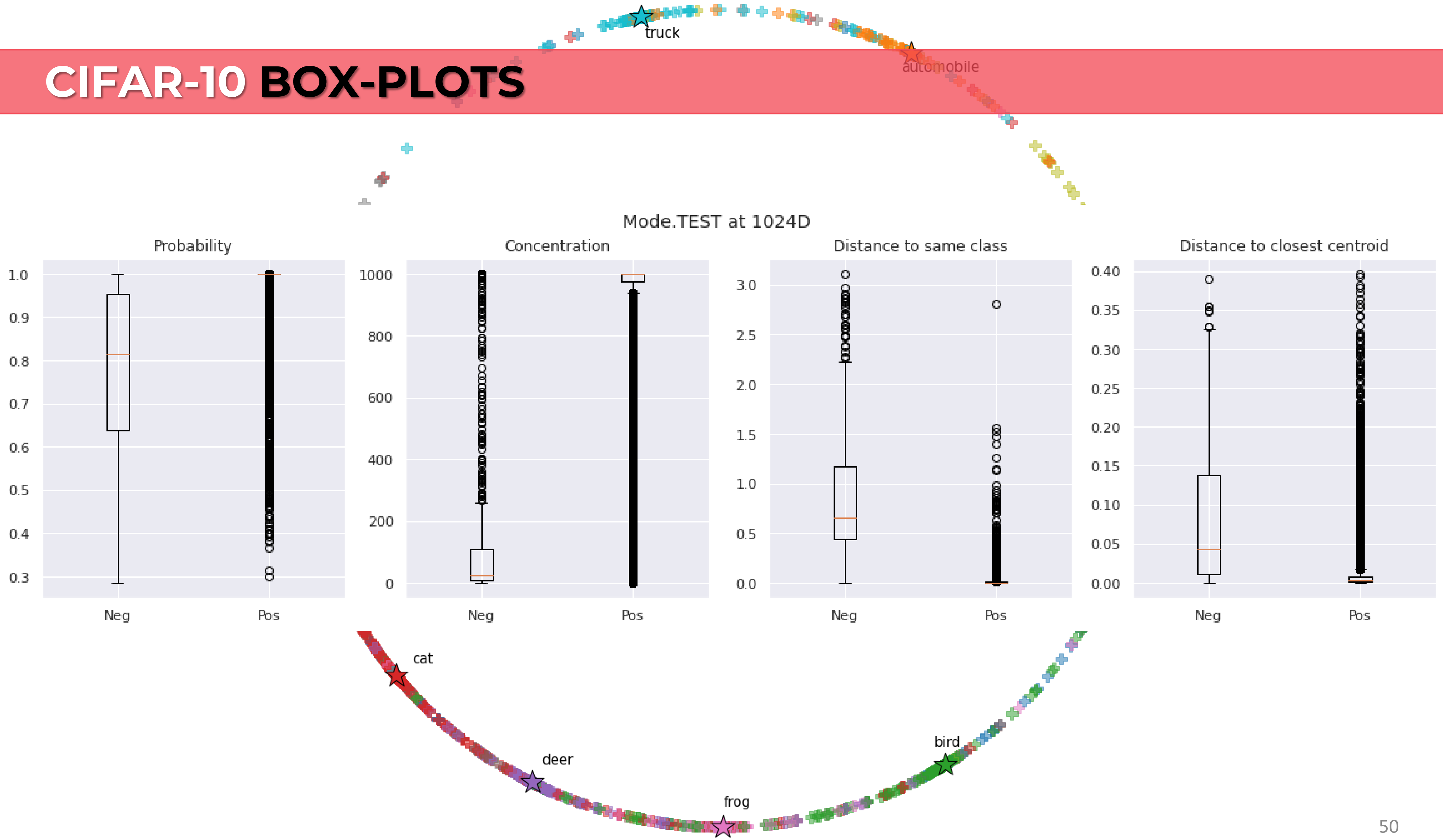

**NCL LOSS**

| | **1024D** | **2D** | **ACC @80%** |
|---|---|---|---|
| TRAIN | 94.8% | 93.4% | |
| EVAL | **93.8%** | **92.2%** | **99.0%** |

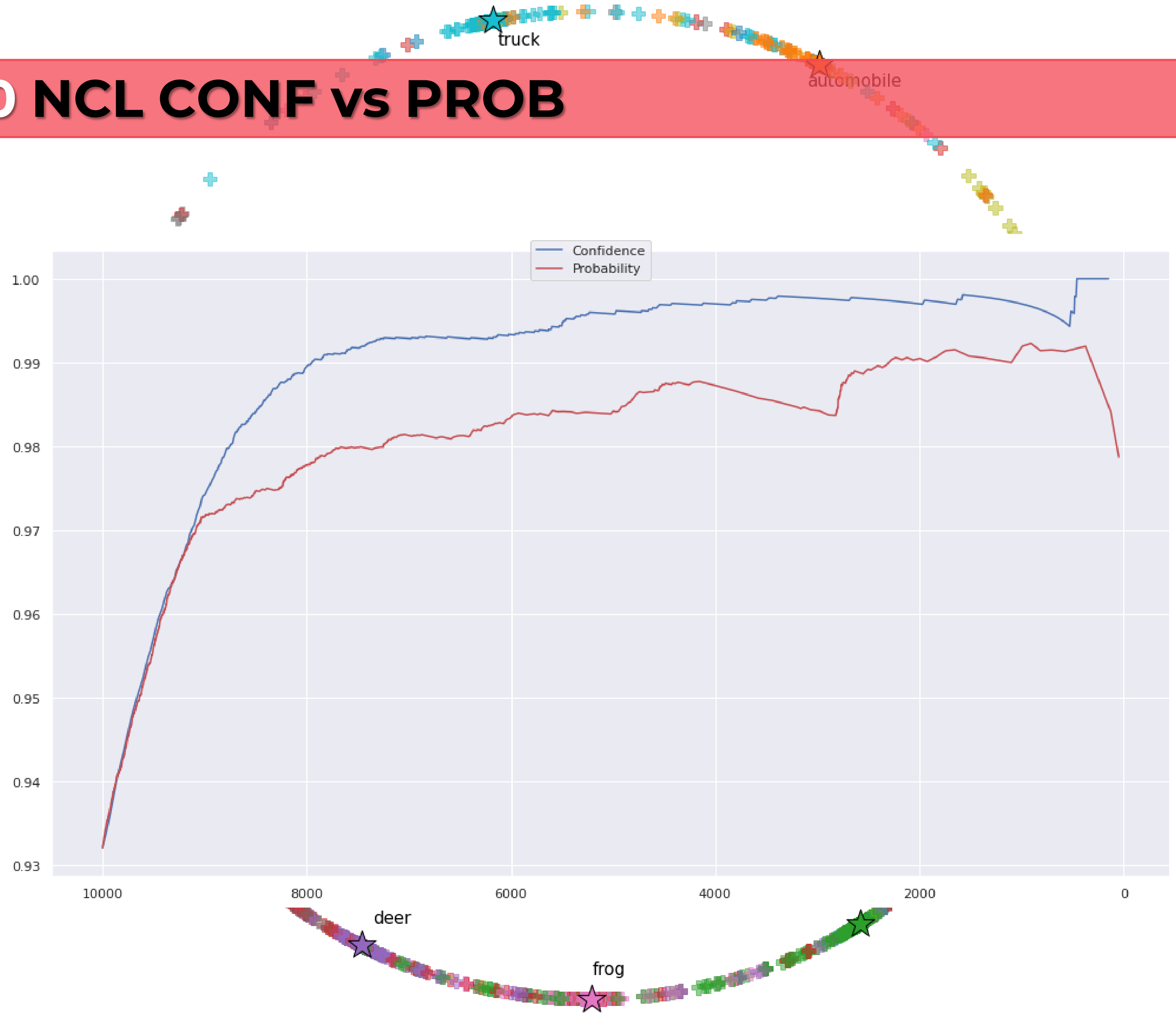
**TRIPLET LOSS**

| | **1024D** | **2D** | **ACC @80%** |
|---|---|---|---|
| TRAIN | 92.8% | 91.4% | |
| EVAL | 91.3% | 90.6% | 96.1% |

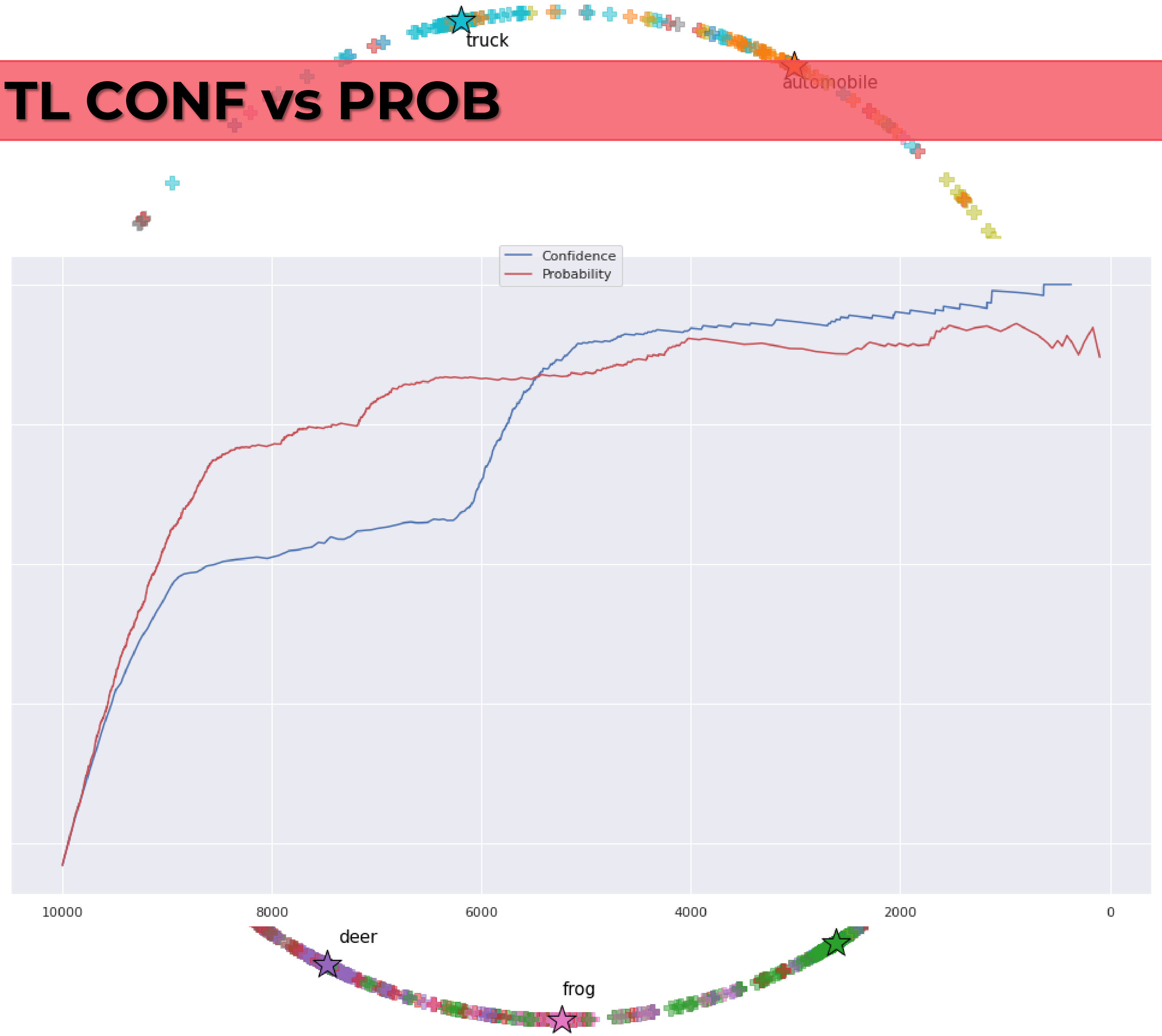
# CIFAR-10 BOX-PLOTS



# CIFAR-10 NCL CONF vs PROB



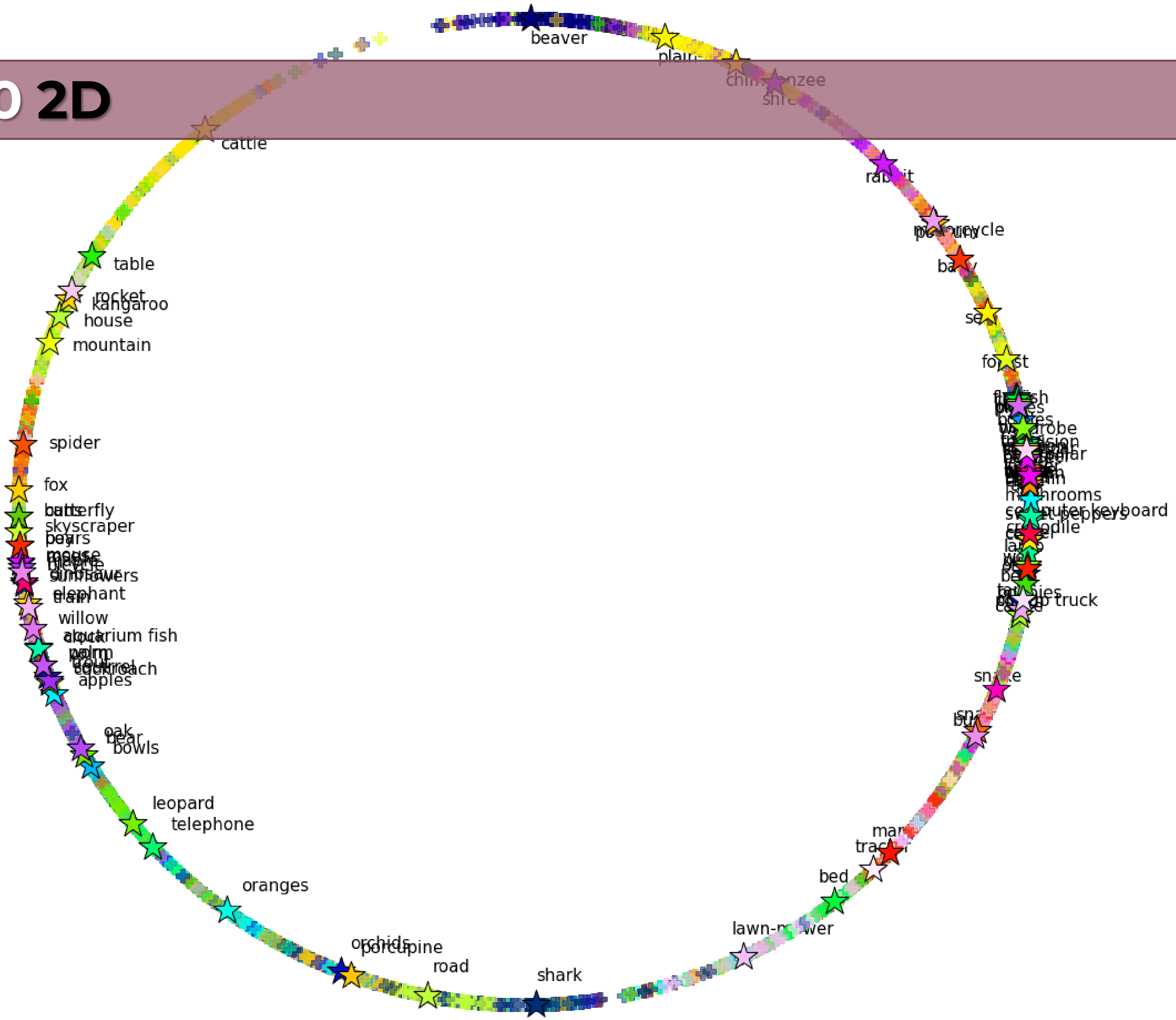
# CIFAR-10 TL CONF vs PROB



# CIFAR-10 RESULTS

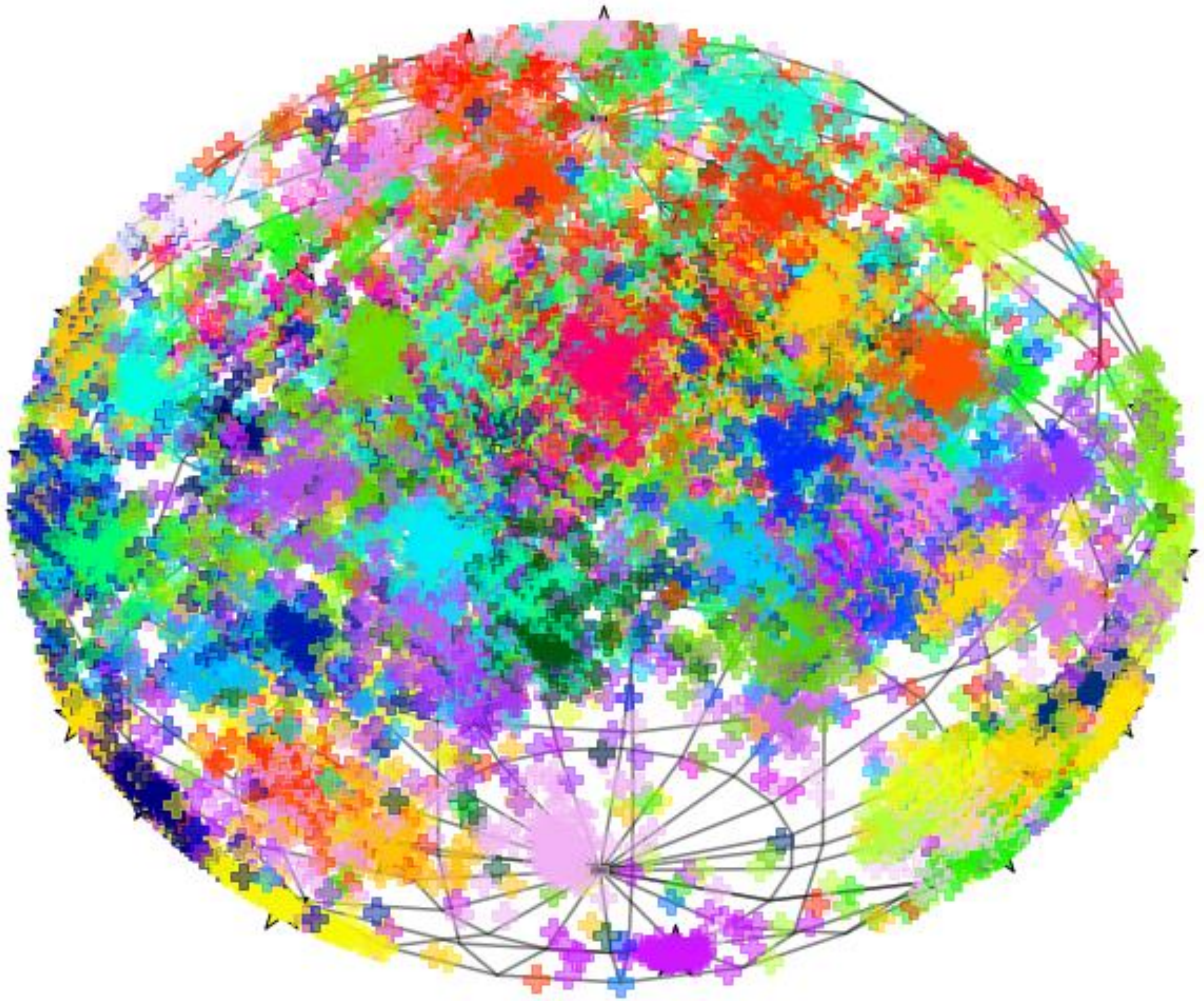
# RESULTS (CIFAR-100)

# CIFAR-100 2D



beaver
cattle
table
rocket
kangaroo
house
mountain
spider
fox
skyscraper
sunflowers
elephant
willow
aquarium fish
cockroach
apples
oak
bowls
leopard
telephone
oranges
orchids
porcupine
road
shark
lawn-mower
bed
snake
pickup truck
computer keyboard
mushrooms
crocodile
forest

# CIFAR-100 3D

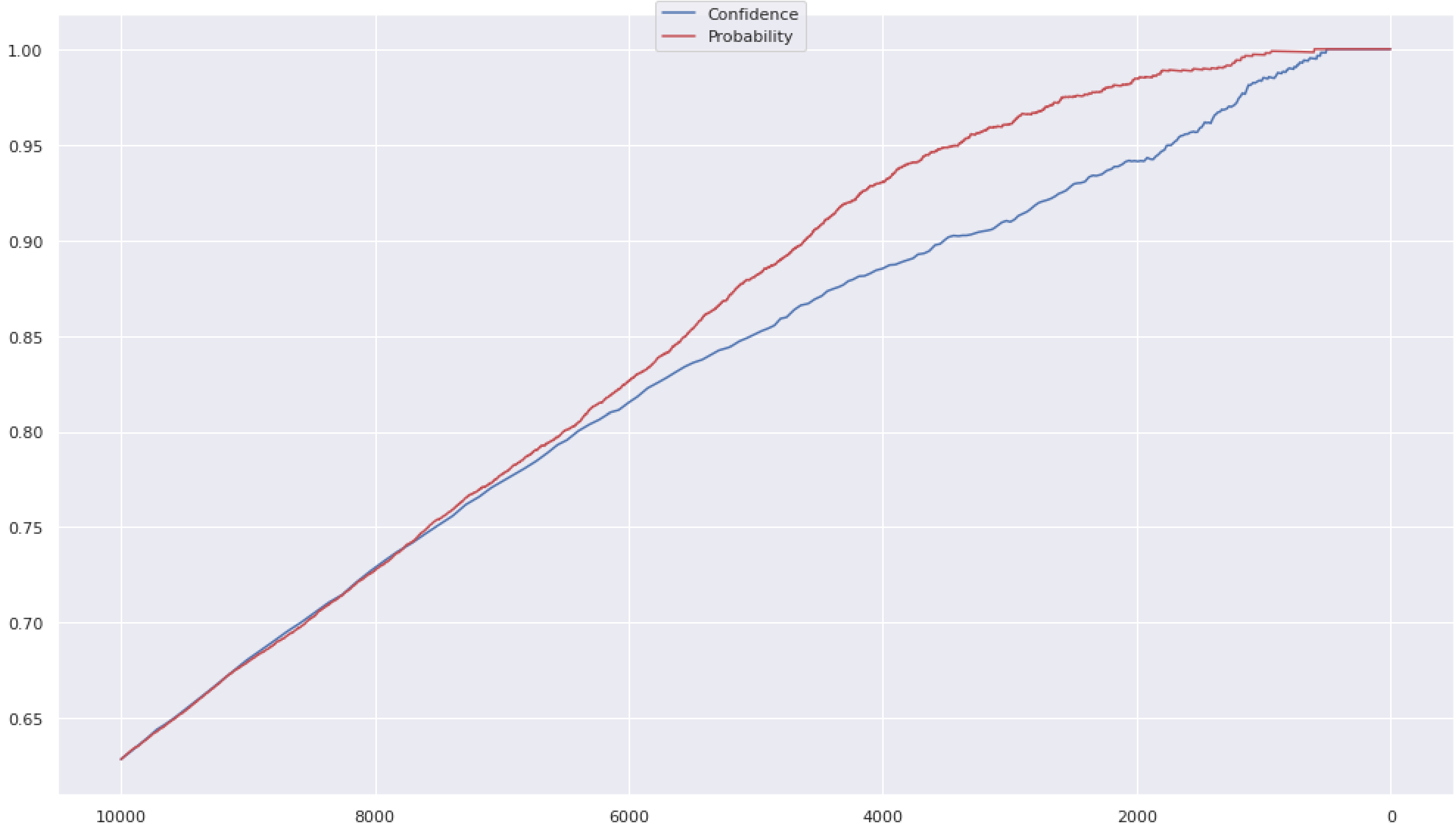


# CIFAR-100 RESULTS

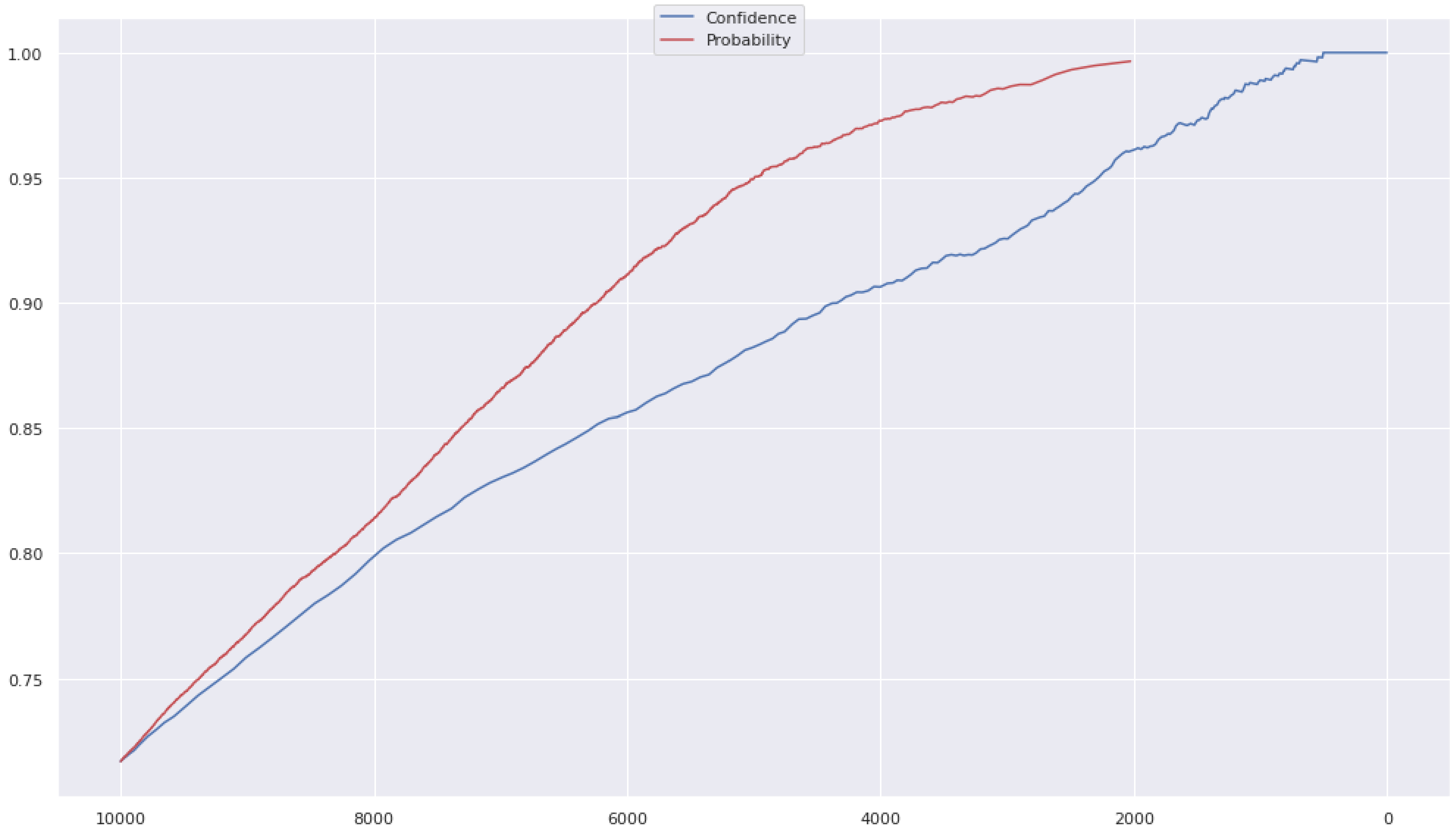
| | 1024D | 2D | 3D | 4D | 5D |
|---|---|---|---|---|---|
| TRAIN | 85.0% | 21.8% | 52.8% | 62.8% | 68.2% |
| EVAL | **71.4**$\pm$**0.35** | 20.2% | 47.2% | 58.6% | **62.6%** |
| EVAL REL. | | - | +27.0 | +11.2 | +4.0 |

| ACC @80% | 1024D (@5D) | 2D | 3D | 4D | 5D |
|---|---|---|---|---|---|
| EVAL | 80.2% | 22.9% | 55.9% | 68.4% | 73.2% |

# CIFAR-100 RESULTS



# CIFAR-100 RESULTS



# CONCLUSIONS

# SUM UP

- NCL overperforms TL
  - TL still wins in Deep Metric Learning
- Can be coupled with Cross Entropy and acts as a regularizer
- Provides confidence estimates
  - OOD detection

# FUTURE STEPS

- Confidence needs calibration
  - $[0,1000] \rightarrow [0,1]$
  - Compare with OOD papers [ECE + Brier]
- Higher dimensionality
  - Would help with CIFAR-100, TinyImagenet, etc.