Capstone Project: Heatmap Anomaly Detection

Week 8 Progress Report

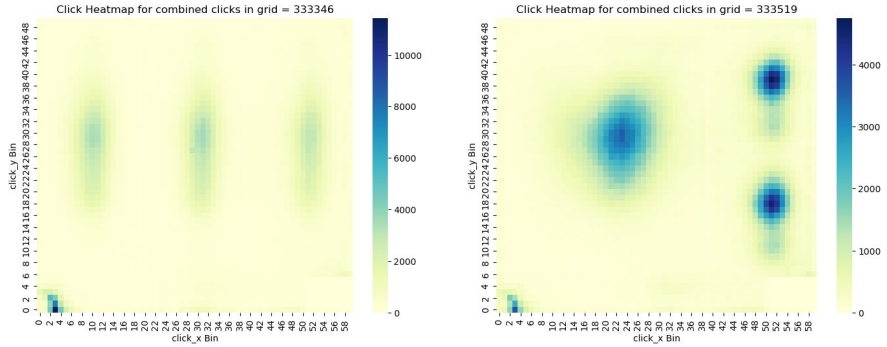
This week:

1. Write report
2. PCA on combined dataset
3. Cosine similarity on pretrained ResNet/ViT
4. Fine-tune on synthetic data
5. Questions

Recap:

Definition of “clearly broken grid:”

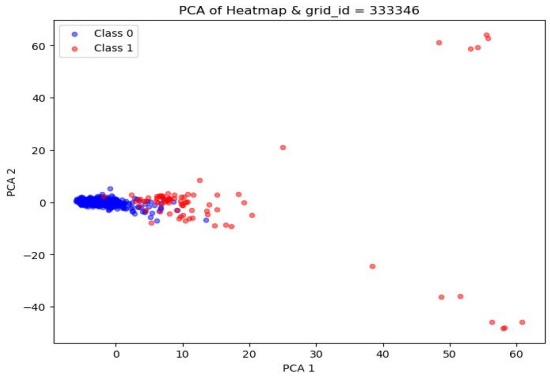
Triangle/line structure not visible even with noisy bootstrap enhancement. We do not care about rest of heatmap as long as this pattern is clearly defined.

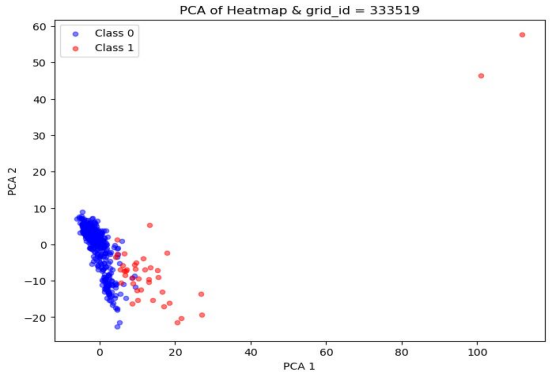


1. Created baseline of “clearly broken banners” for two grids:
   - “Triangle grid”: 57 clearly broken out of 872 → **~6.5%**
   - “Line grid”: 113 clearly broken out of 861 → **~13.5%**
   - Is it reasonable that there is such a discrepancy?

2. Added heatmap images for banners classified as “broken” to GitHub.

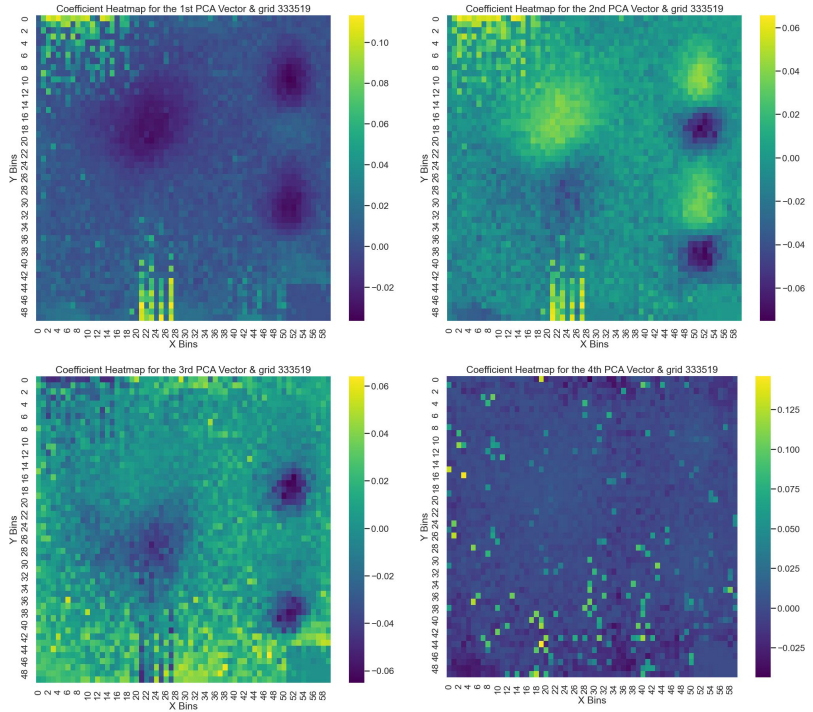
PCA (recap):





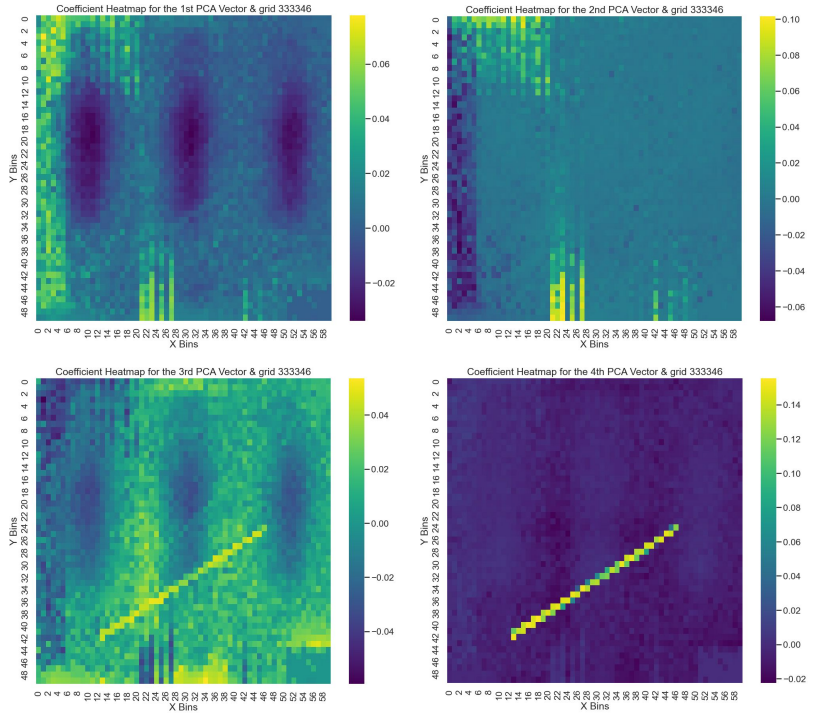
- PCA on 3000 dimensional binned heatmap vectors (EV ~ 4%, 3% resp.)
- Shows quite good pattern separating broken/non-broken banners.
- PCA vectors seem to be transferable between grid_id's (at least with 3 banners).

Analysis of PCA vectors (333519):



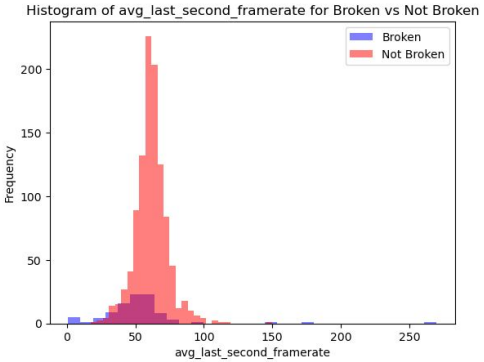
- Analysis of PCA-basis vectors:
  - PCA_1 = c_{ij} bucket_{ij}, etc.
  - Draw heatmap for c_{ij} coefficients.
- PCA1:
  - Captures “noise” outside of main clusters.
  - Especially focuses on two regions on the banner-border
- PCA2:
  - Focuses more on central clusters again together with similar outside regions.
- PCA3:
  - Similar to PCA1.
- PCA4:
  - Not really sure.

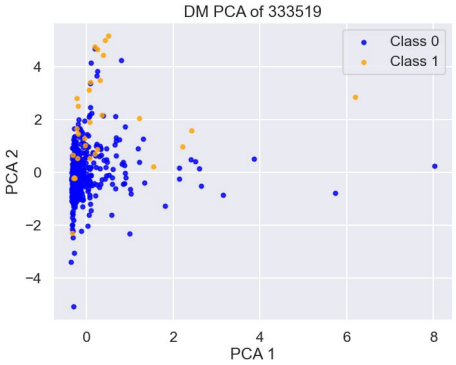
Analysis of PCA vectors (333346):

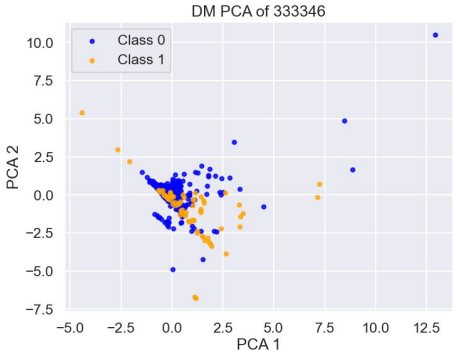


- PCA1:
  - Again, captures “noise” outside of main clusters.
  - Stronger focus on left border.
  - Similar structure on top and bottom.
- PCA2:
  - Again similar structure on top and bottom → not really sure why this is grid_id-independent
- PCA3:
  - Again similar to PCA1.
  - Including “bot-like” behavior
- PCA4:
  - Captures “bot-like” behavior.

Combine datasets (Recap)

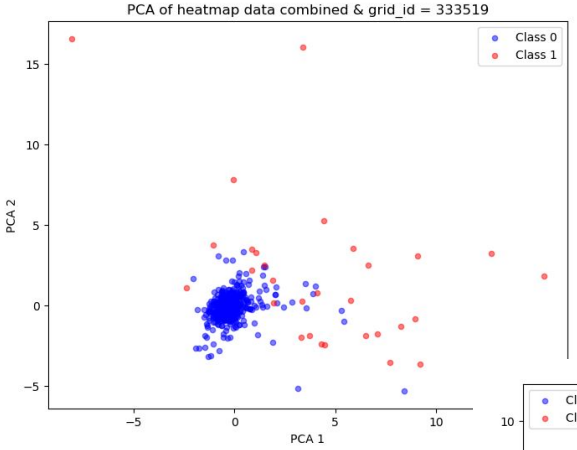


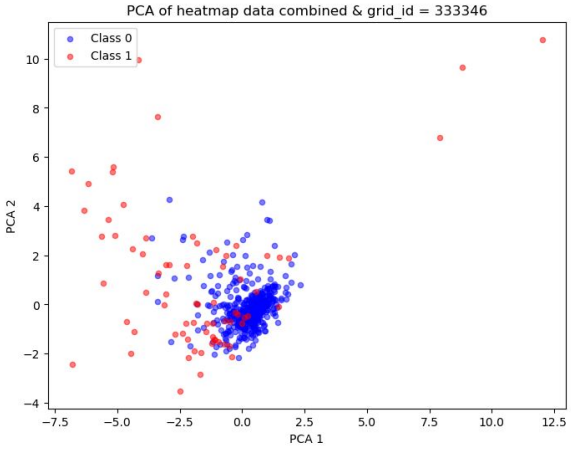




- Basic supervised classifiers (Decision Trees, Random Forest, Logistic Regression, XGBoost, etc) don't perform well
  - Problem with **unbalanced dataset** and overfitting.
    - **Upsample/downsample** → slight improvements but not competitive
  - **Overfitting** → Grid search over model complexity (regularization).
  - Add interactions/higher-order features
- **To Do** → Try to improve using SMOTE/other more sophisticated "upsamplying" strategies.
- PCA less powerful but shows some structure.

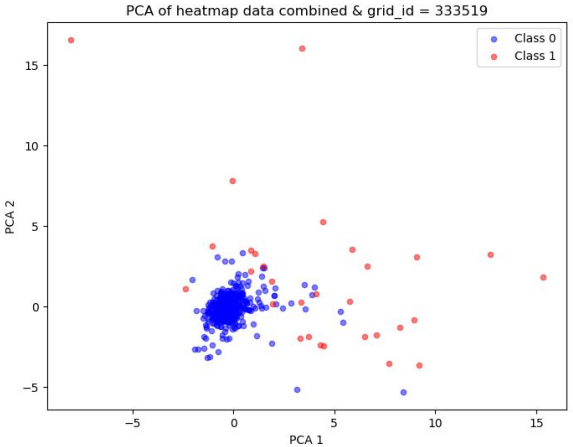
Combined Datasets

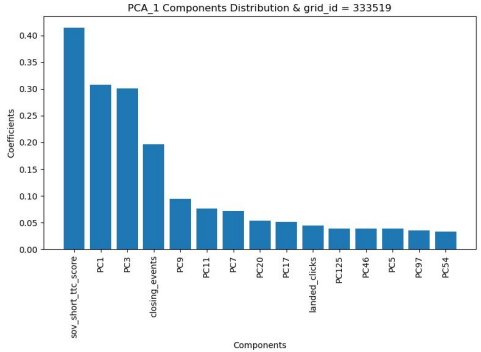


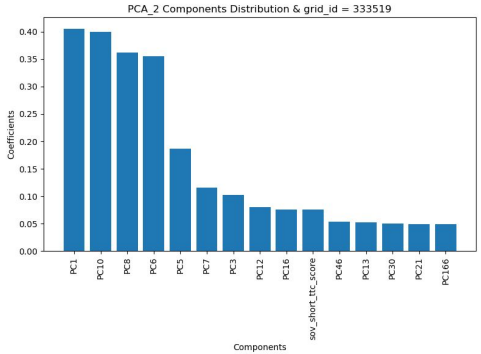


- Different methods of aggregating PCA's:
  - PCA on both individually → combine → scale → combined PCA (top)
  - PCA on HM only → combine → scale → combined PCA (bottom)
  - Some dependence on nr of heatmap-PCA components.
- Adding more components changes scores a little bit.

Combined Datasets

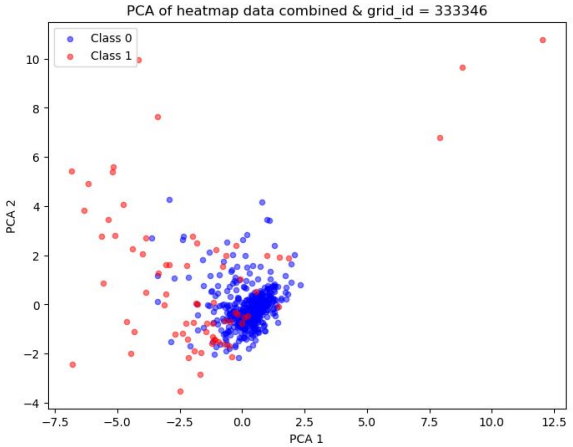


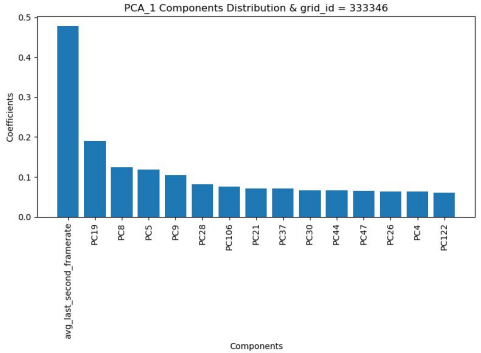


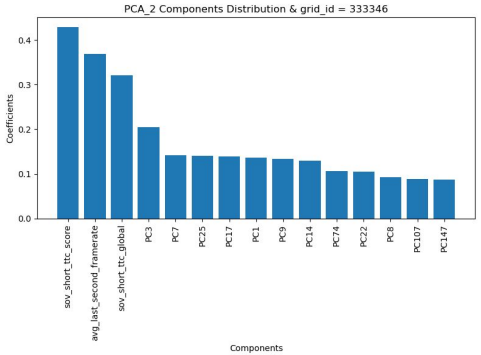


- Metrics component sov_short_ttc_score plays important role in 1st aggregated PCA component + other lower PCA's from heatmap and closing_event.
- 2nd PCA mostly contains PCA components from heatmap.
- Crucial combination of the two datasets and seems to improve on heatmap-only clustering.

Combined Datasets

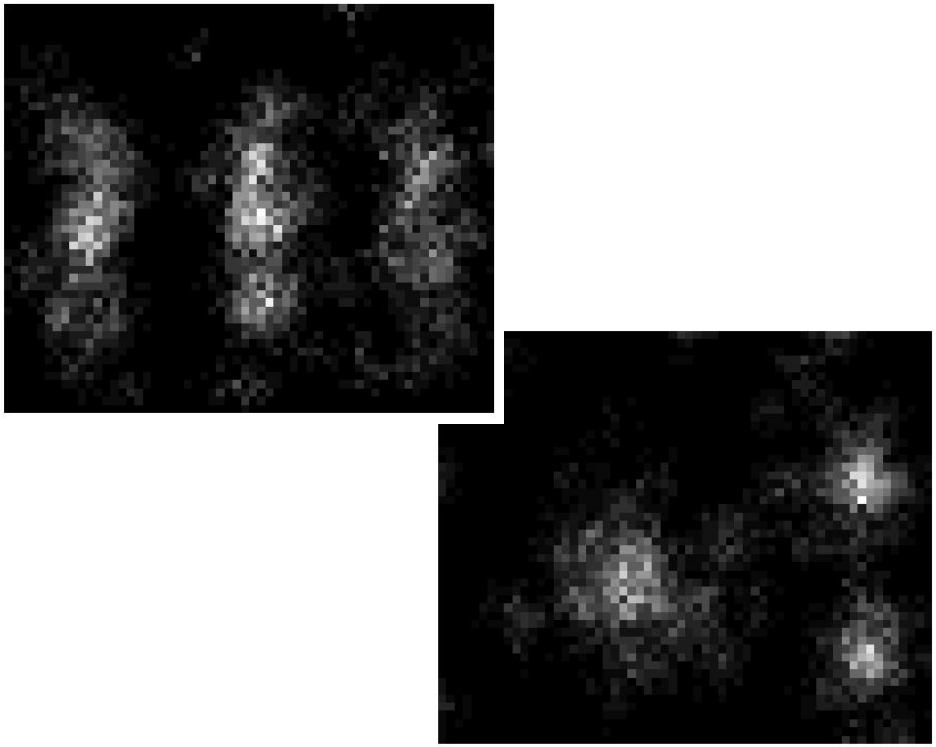






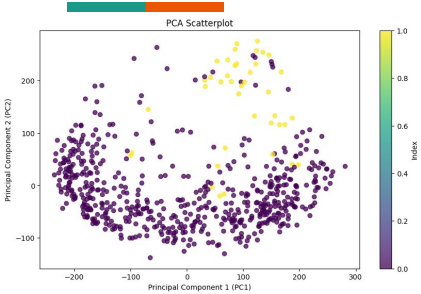
- For this grid_id, avg_last_second_framerate seems to be extremely important for 1st PCA component.
- 2nd PCA component contains other metric dataset ingredients.

Pretrained ViT/ResNet (recap):

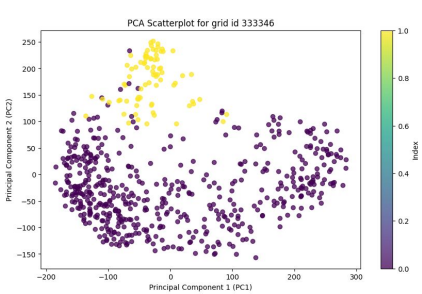


- Feed (transformed and binned) heatmaps into pre-trained ViT/ResNeT.
  - google/ViT: Transformer-based architecture, 14M images (224x224), 21k classes
  - Microsoft/ResNet-1k: trained on ImageNet-1k (224x224), 1k classes.
- Extract features (before classification head)
  - ViT → 151296 dim'l feature vector
  - ResNet → 2048 dim'l feature vector
- Play with upsampling (bootstrapping + noise)
- PCA and other dim'l reduction techniques
  - Apply clustering methods

ViT + PCA:

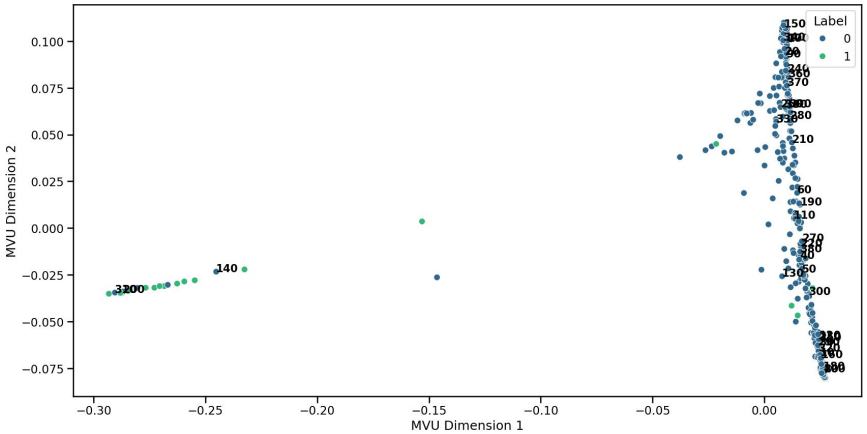


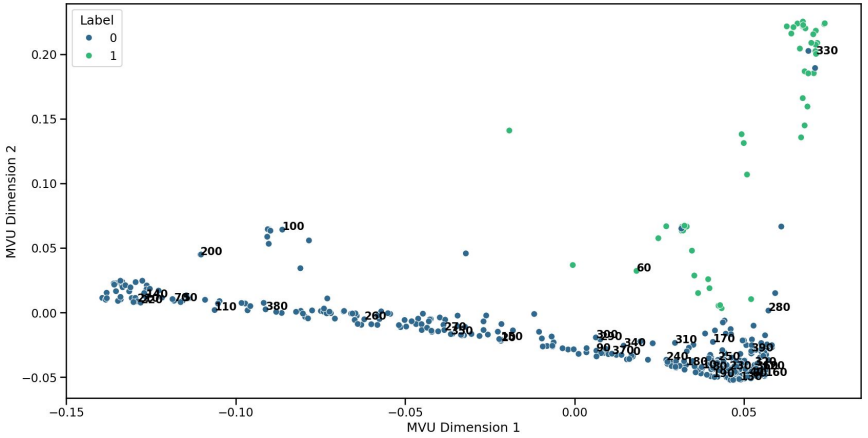
Grid 333510, EV ~ 18%



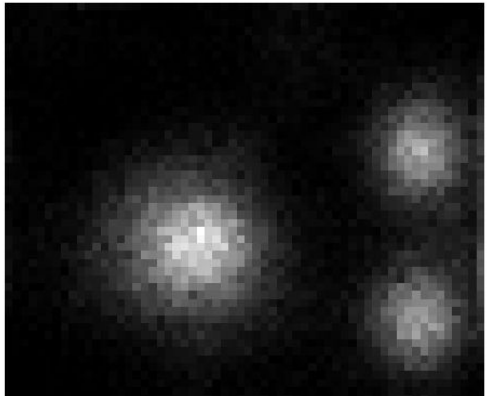
Grid 333346, EV ~ 17%

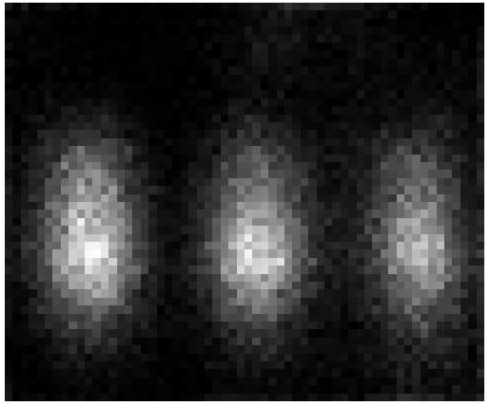
- PCA → some structure, but not as good as vanilla method.
- (nonlinear) MVU shows clear patterns however → clearly features contain important information.
- Clustering on PCA's not performant.
- → require fine-tuning? Contrastive learning on synthetic data?

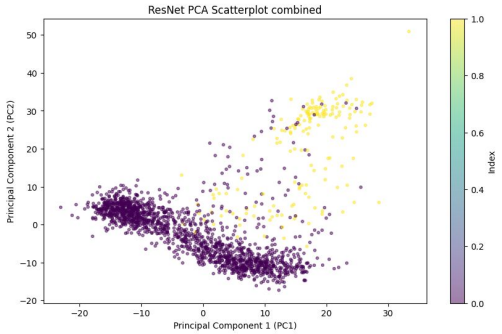


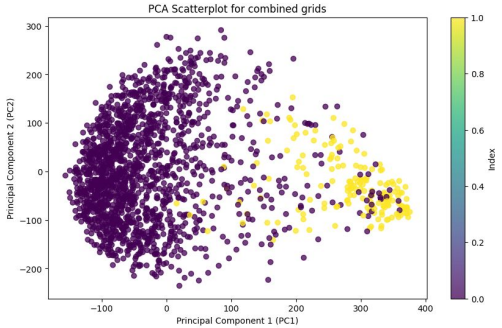


Data enhancement:

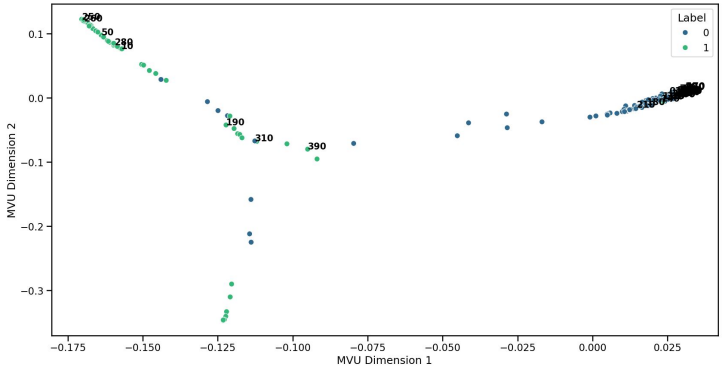




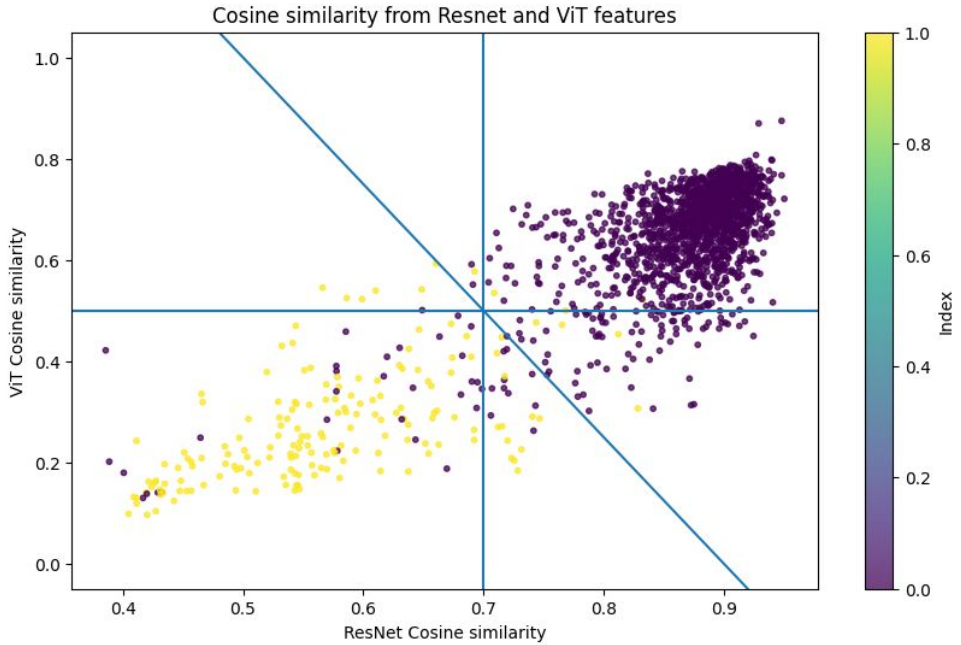




- We now bootstrap upsample to 50’000 clicks per image.
- PCA’s look a lot more promising.
- MVU also look very indicative of deep structure from pre-trained models.
- => continue with enhanced data from now on.

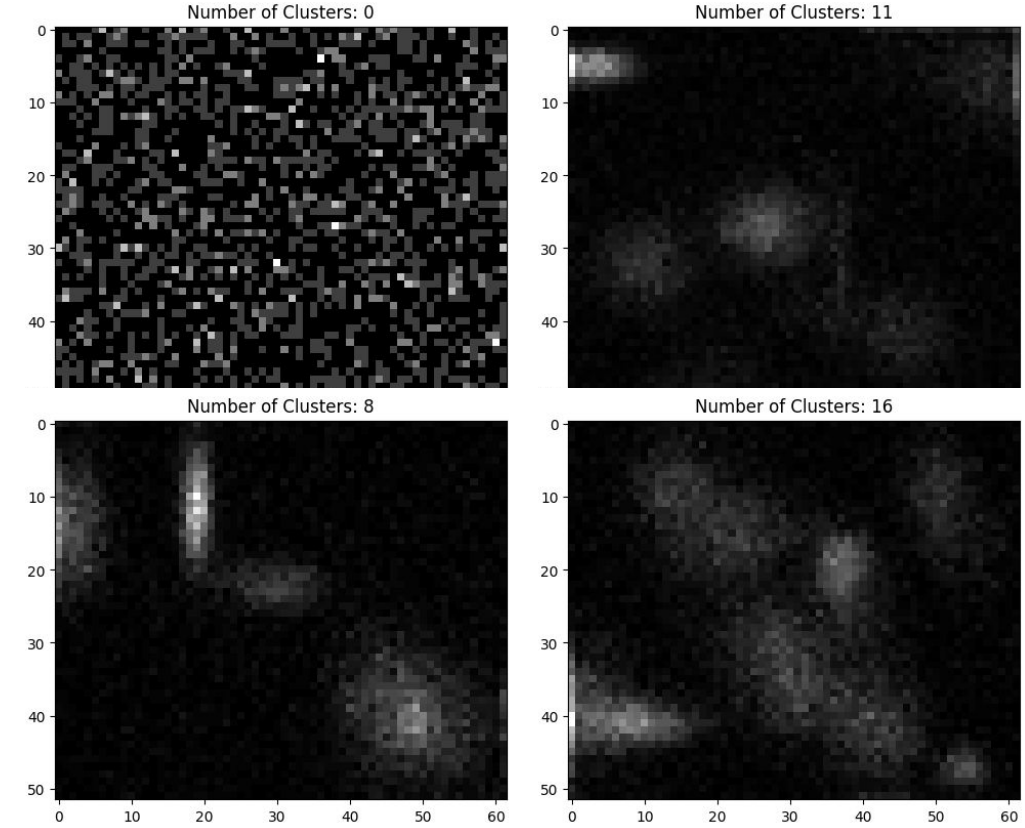


Cosine similarity



- PCA → gives some intuition, but very limited
- MVU → shows there is underlying structure.
- Feature vectors usually evaluated using **cosine similarity** (angle between vectors in high-dimensional feature space)
- **Cosine similarity** has actual meaning (while PCA does not really, since “maximum variance” is inherently imprecise).
- Draw picture of cosine similarity using ResNet and ViT feature vectors.
- What choice of “anker”?
  - Pick random choice
  - Pick average non-broken grid_id’s.

Fine-tune ResNet-50



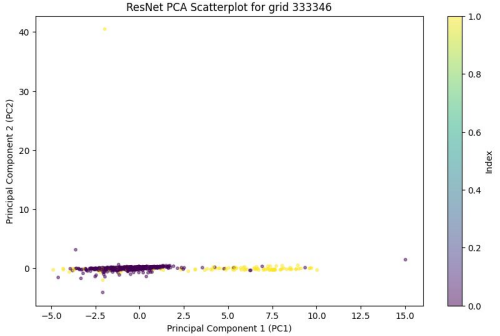
- Fine-tune feature vectors of ResNet-50 architecture:
  - Pretrained ResNet
  - Generate synthetic images:
    - Random # of clusters (0-20)
    - Random center for clusters
    - Random covariance for Gaussians
    - Random # of clicks/cluster
    - Overlay random noise
  - Add linear classification head (softmax)
  - “Learn” detection of number of clusters (cross-entropy loss).

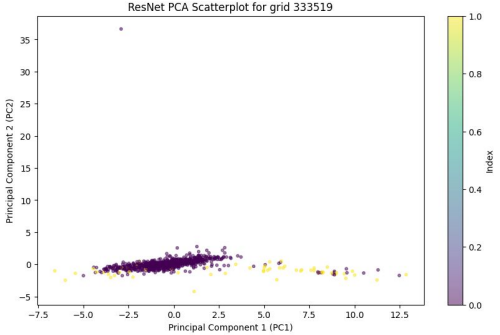
Fine-tune ResNet-50

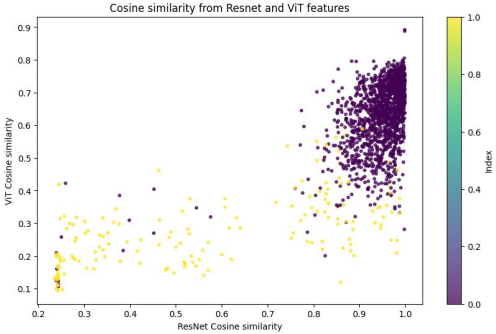
```
Epoch [26/50], Loss: 1.3955528624355793
Epoch [26/50], Validation Loss: 1.3435876028878349, Validation Accuracy: 0.45
Epoch [27/50], Loss: 1.3637148477137089
Epoch [27/50], Validation Loss: 1.4837983335767473, Validation Accuracy: 0.43
Epoch [28/50], Loss: 1.3805490285158157
Epoch [28/50], Validation Loss: 1.238404597554888, Validation Accuracy: 0.45
Epoch [29/50], Loss: 1.3897866569459438
Epoch [29/50], Validation Loss: 1.4044449925422668, Validation Accuracy: 0.375
Epoch [30/50], Loss: 1.3966036960482597
Epoch [30/50], Validation Loss: 1.3146330118179321, Validation Accuracy: 0.475
Epoch [31/50], Loss: 1.39871409162879
Epoch [31/50], Validation Loss: 1.3485074213572912, Validation Accuracy: 0.425
Epoch [32/50], Loss: 1.3727771192789078
Epoch [32/50], Validation Loss: 1.4909079245158605, Validation Accuracy: 0.405
Epoch [33/50], Loss: 1.460495449602604
Epoch [33/50], Validation Loss: 1.3042080572673254, Validation Accuracy: 0.44
Epoch [34/50], Loss: 1.4012102782726288
Epoch [34/50], Validation Loss: 1.3243705545152937, Validation Accuracy: 0.4
Epoch [35/50], Loss: 1.3832057006657124
Epoch [35/50], Validation Loss: 1.263480544090271, Validation Accuracy: 0.475
Epoch [36/50], Loss: 1.386552106589079
Epoch [36/50], Validation Loss: 1.3199748992919922, Validation Accuracy: 0.48
Epoch [37/50], Loss: 1.3840462379157543
Epoch [37/50], Validation Loss: 1.4264745201383318, Validation Accuracy: 0.46
Epoch [38/50], Loss: 1.4330132870442867
Epoch [38/50], Validation Loss: 1.4097107137952531, Validation Accuracy: 0.435
Epoch [39/50], Loss: 1.3397572189569473
Epoch [39/50], Validation Loss: 1.565937944820949, Validation Accuracy: 0.33
Epoch [40/50], Loss: 1.3547335863113403
Epoch [40/50], Validation Loss: 1.426804883139474, Validation Accuracy: 0.405
Epoch [41/50], Loss: 1.3820106461644173
Epoch [41/50], Validation Loss: 1.459223440715245, Validation Accuracy: 0.4
Epoch [42/50], Loss: 1.3851871229708195
Epoch [42/50], Validation Loss: 1.4127788032804216, Validation Accuracy: 0.4
Epoch [43/50], Loss: 1.3986276760697365
Epoch [43/50], Validation Loss: 1.3964507750102453, Validation Accuracy: 0.44
Epoch [44/50], Loss: 1.4363364800810814
Epoch [44/50], Validation Loss: 1.38430084500994, Validation Accuracy: 0.41
Epoch [45/50], Loss: 1.3677100390195847
Epoch [45/50], Validation Loss: 1.360924584524972, Validation Accuracy: 0.415
Epoch [46/50], Loss: 1.3937365859746933
Epoch [46/50], Validation Loss: 2.652651412146432, Validation Accuracy: 0.29
Epoch [47/50], Loss: 1.4235710762441158
Epoch [47/50], Validation Loss: 1.2446696502821786, Validation Accuracy: 0.47
Epoch [48/50], Loss: 1.438164971768856
Epoch [48/50], Validation Loss: 1.5681287050247192, Validation Accuracy: 0.4
Epoch [49/50], Loss: 1.3514236211776733
Epoch [49/50], Validation Loss: 1.3991124289376395, Validation Accuracy: 0.45
Epoch [50/50], Loss: 1.3968842066824436
Epoch [50/50], Validation Loss: 1.3249718121119909, Validation Accuracy: 0.45
```

- Fine-tune feature vectors of ResNet-50 architecture:
  - Pretrained ResNet
  - Generate synthetic images:
    - Random # of clusters (0-20)
    - Random center for clusters
    - Random covariance for Gaussians
    - Random # of clicks/cluster
    - Overlay random noise
  - Add linear classification head (softmax)
  - “Learn” detection of number of clusters (cross-entropy loss).
- After some training get ~ 40% accuracy on predicting # of clusters from a given image (somewhat depending on some of the hyperparameters).

Fine-tune ResNet-50

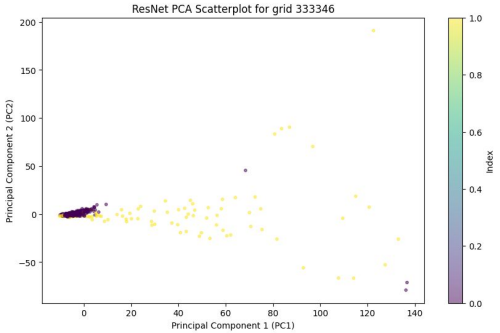


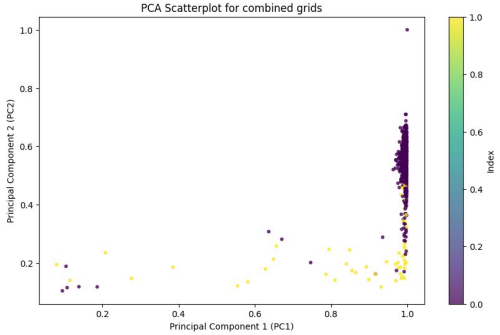


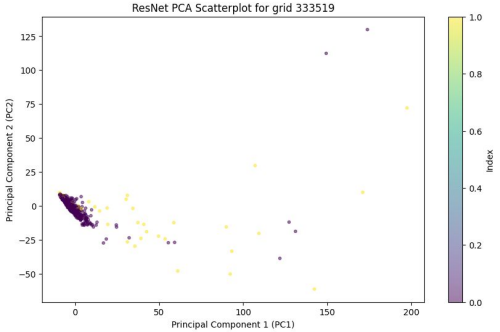


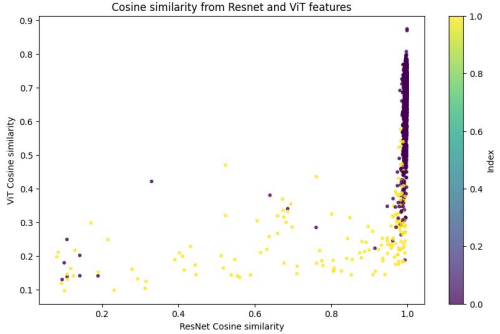
- PCA results are very interesting (need to understand better)
- Cosine similarity shows a clear distinction between broken and non-broken banner in the ResNet-direction.
- Dependent on training inputs (especially the max extent/covariance of the clusters and the number of max clicks per cluster/noise).

Fine-tune ResNet-50 (more training)





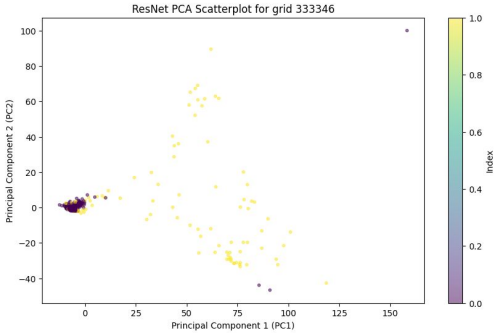


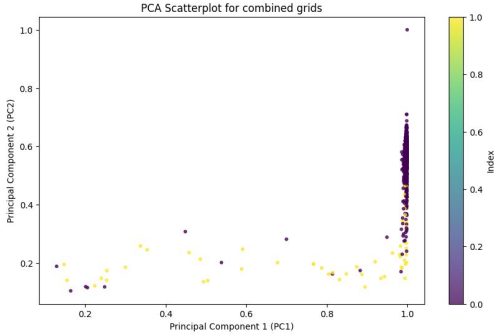


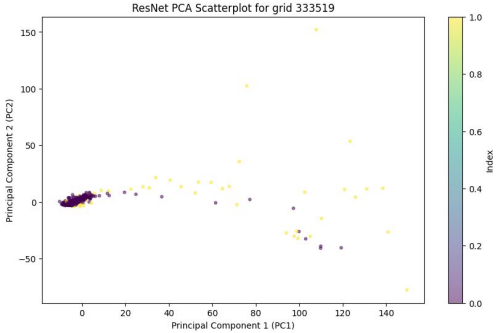
- PCA results are very interesting (need to understand better)
- Cosine similarity shows a clear distinction between broken and non-broken banner in the ResNet-direction.
- Dependent on training inputs (especially the max extent/covariance of the clusters and the number of max clicks per cluster/noise).

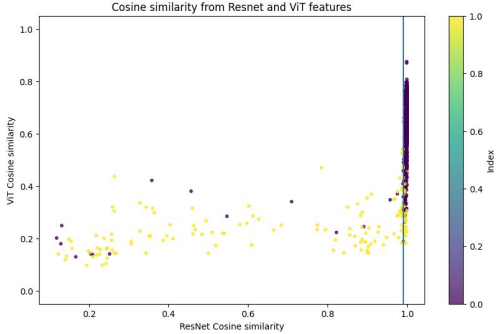
```
Epoch [288/300], Loss: 1.4082686007022858
Epoch [288/300], Validation Loss: 1.494951946394784, Validation Accuracy: 0.44
Epoch [289/300], Loss: 1.3965076766908169
Epoch [289/300], Validation Loss: 1.335465567452567, Validation Accuracy: 0.445
Epoch [290/300], Loss: 1.4133847169578075
Epoch [290/300], Validation Loss: 2.138043795313154, Validation Accuracy: 0.34
Epoch [291/300], Loss: 1.4208088926970959
Epoch [291/300], Validation Loss: 6.254529067448208, Validation Accuracy: 0.235
Epoch [292/300], Loss: 1.4183865077793598
Epoch [292/300], Validation Loss: 1.7257601022720337, Validation Accuracy: 0.315
Epoch [293/300], Loss: 1.3947353065013885
Epoch [293/300], Validation Loss: 1.4097038166863578, Validation Accuracy: 0.42
Epoch [294/300], Loss: 1.3900279067456722
Epoch [294/300], Validation Loss: 1.5141012838908605, Validation Accuracy: 0.37
Epoch [295/300], Loss: 1.3473711051046848
Epoch [295/300], Validation Loss: 1.9365281888416834, Validation Accuracy: 0.345
Epoch [296/300], Loss: 1.357010602951049 8
Epoch [296/300], Validation Loss: 1.663163764136178, Validation Accuracy: 0.345
Epoch [297/300], Loss: 1.405576653778553
Epoch [297/300], Validation Loss: 1.4058324950081962, Validation Accuracy: 0.42
Epoch [298/300], Loss: 1.3370253331959248
Epoch [298/300], Validation Loss: 1.5712675877979823, Validation Accuracy: 0.36
Epoch [299/300], Loss: 1.3868932351469994
Epoch [299/300], Validation Loss: 1.2801916088376726, Validation Accuracy: 0.485
Epoch [300/300], Loss: 1.3897702284157276
Epoch [300/300], Validation Loss: 1.4677356311253138, Validation Accuracy: 0.39
```

Fine-tune ResNet-50 (even more training)





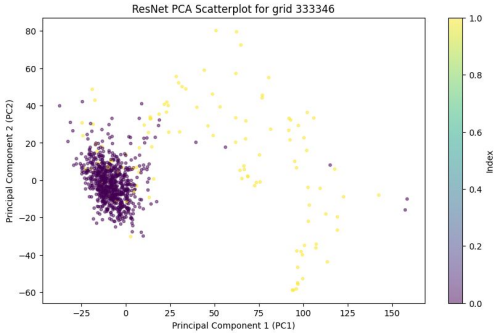


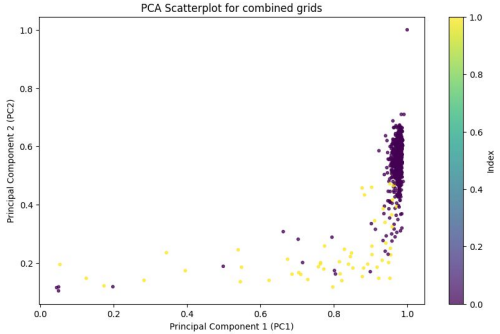


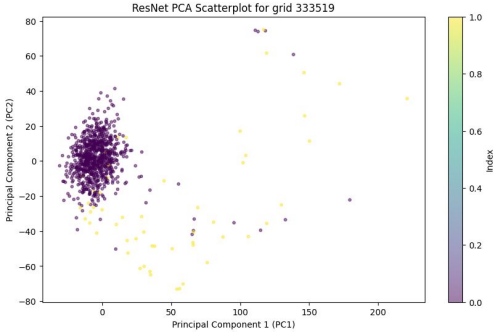
- PCA results are very interesting (need to understand better)
- Cosine similarity shows a clear distinction between broken and non-broken banner in the ResNet-direction.
- Dependent on training inputs (especially the max extent/covariance of the clusters and the number of max clicks per cluster/noise).

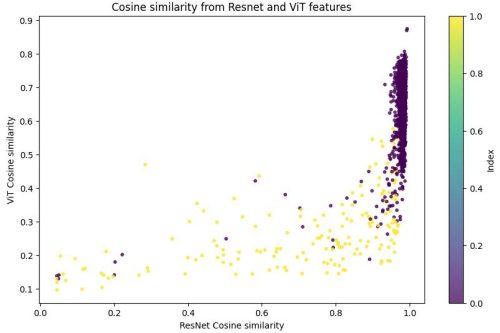
```
Epoch [108/300], Loss: 1.3165524266660213
Epoch [108/300], Validation Loss: 1.3463102749415807, Validation Accuracy: 0.42
Epoch [109/300], Loss: 1.3854121789336205
Epoch [109/300], Validation Loss: 2.1799562658582414, Validation Accuracy: 0.35
Epoch [110/300], Loss: 1.3949197046458721
Epoch [110/300], Validation Loss: 2.1274535826274326, Validation Accuracy: 0.305
Epoch [111/300], Loss: 1.36465635150671
Epoch [111/300], Validation Loss: 1.2870481354849679, Validation Accuracy: 0.475
Epoch [112/300], Loss: 1.3625223636627197
Epoch [112/300], Validation Loss: 1.969708936555045, Validation Accuracy: 0.365
Epoch [113/300], Loss: 1.3515837192535 4
Epoch [113/300], Validation Loss: 1.2907083800860815, Validation Accuracy: 0.46
Epoch [114/300], Loss: 1.299885120242834
Epoch [114/300], Validation Loss: 1.5937663657324654, Validation Accuracy: 0.395
Epoch [115/300], Loss: 1.3517960645258427
Epoch [115/300], Validation Loss: 2.5957248210906982, Validation Accuracy: 0.345
Epoch [116/300], Loss: 1.4256849214434624
Epoch [116/300], Validation Loss: 1.8909719671521867, Validation Accuracy: 0.345
Epoch [117/300], Loss: 1.3820858616381884
Epoch [117/300], Validation Loss: 1.358289701598031, Validation Accuracy: 0.435
Epoch [118/300], Loss: 1.373707059770822 5
Epoch [118/300], Validation Loss: 1.2613064391272408, Validation Accuracy: 0.48
Epoch [119/300], Loss: 1.3527532406511295
Epoch [119/300], Validation Loss: 1.3475025721958704, Validation Accuracy: 0.445
Epoch [120/300], Loss: 1.3494292125105858
Epoch [120/300], Validation Loss: 3.494121483394078, Validation Accuracy: 0.29
```

Predict cluster center (preliminary results)









- Predict location of cluster centers now, instead of number of clusters.
- Max 5 clusters (if fewer, predict (-1,-1)) and predict approximate center of cluster.
- Use MSE of distance between prediction and cluster. (Notice need to be careful about ordering of cluster centers → chamfer loss, currently running).

Questions:

- Can we quantify the loss per day(?) arising from a broken banner?
  - How important for the business is this detection
  - What is their current system (very broadly)?
- What should we focus on?
  - Apply fancier methods (contrastive learning, etc)?
  - Productionalize method?
- Is there a way to find out about state of the art methods
  - We couldn't really find anything that is comparable to this project.
- What performance are we expected to provide for zero-shot learning?

Questions: Metrics performance

Aberrant metrics performance? Did not manage to find a clear response on this subject so here are some guesses:

Client permissions

- Ad blockers: incurred the loss of 30% of total industry revenue in 2015 [source]
    - In this case, the broken banners could essentially be random clicks?
- Rejection of third party cookies (70% of individuals, UK survey)

Technical issues

- Wrong display image?
- Display error vs link error
- Advertisement being paused, removed or disapproved [source]

Other potential aberrances

- Landing page irrelevant?
- In practice: resetting ad ID helps ads to load (reason unclear)

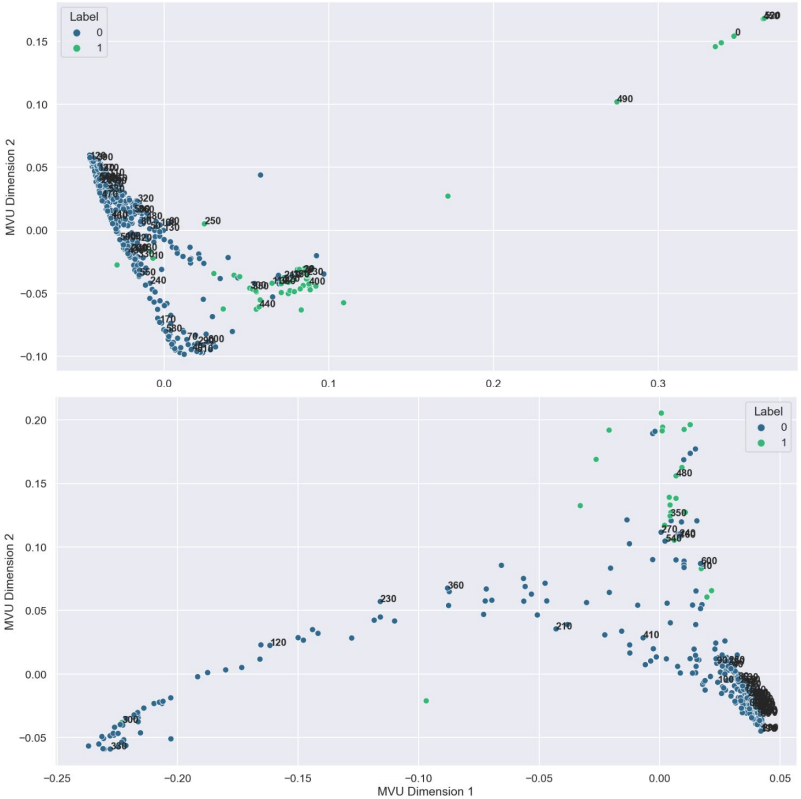
Questions: How is click data collected?

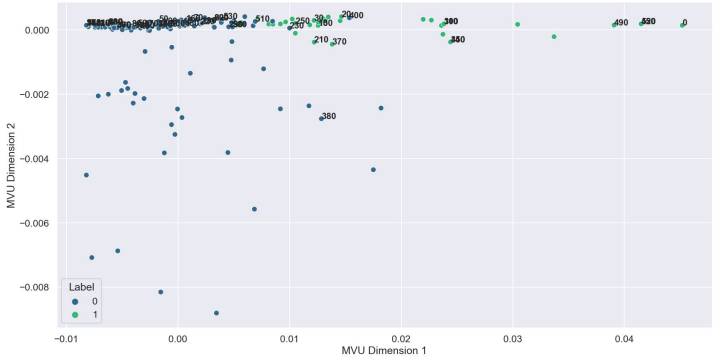
Next steps:

- Further exploration of “basic models”:
  - Stability of models across grids.
    - Access to more grid’s?
  - Can we extend our basic clustering models to do zero-shot learning?
- “Productionize” current models into single pipeline:
  - Clustering on heatmap
  - Clustering on Metrics+heatmap
  - Click clustering methods
  - Clustering of pretrained features (?)
  - → combine into Majority voting pipeline → stability?
- Does fine-tuning actually help?
  - What is a good set of parameters to use for synthetic image generation?
  - Is there a better training task? → for example determine the center of each clusters → could lead to better features.
- Combine ViT/ResNet features with vanilla features (PCA, etc)?
- Other models trained on contrastive tasks?
  - Train Autoencoder on actual heatmaps/arbitrary synthetic clusterings (ask AE to recreate original image with discriminative loss).

Appendix

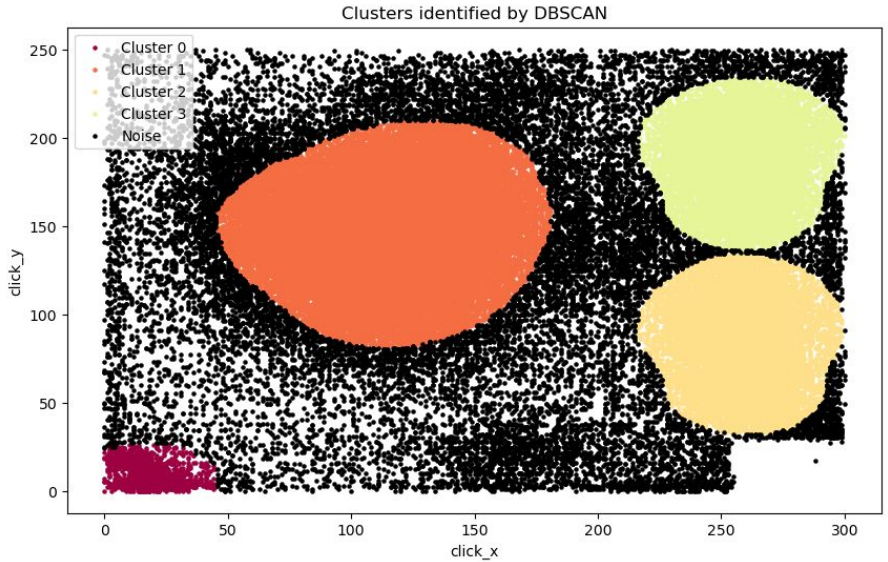
MVU – study “data manifold”:





- Study data manifold using Maximum Variance Unfolding/Semidefinite Embedding (MVU)
  - Intuition: create graph of close points in high dimensional space with distances
  - Use convex optimization to maximize distance between disconnected points s.t. connected points being nearby.

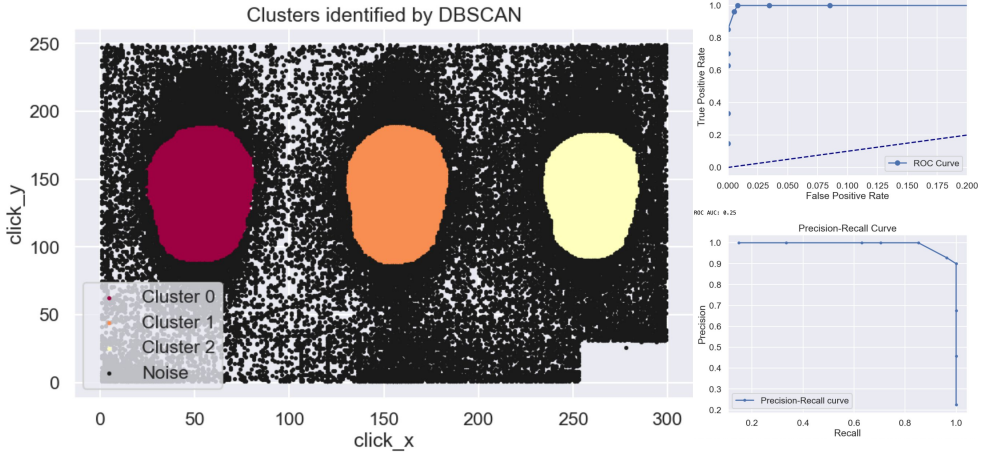
Click clustering method (recap):



1. Bootstrap 100’000 clicks from fully aggregated dataset (filtered by grid_id).
2. Normalize (Standardize)
3. Run DBSCAN cluster with eps = .2 and min_samples = 1000
   → 4 clusters + noise.
4. For given (noisy bootstrap enhanced) domain, get 1-nn for each click in training data and select that label {0,1,2,3}
   a. If pct of points labelled as noise above a certain threshold → anomalous.
   b. Hypothesis testing: p_0 = pct of noise points in training data. H_0: p_0 < noise/total, H_A: p_0 >= noise/total → p-value larger than threshold (cannot reject null) → anomalous.

Click clustering method:

```
Epsilon: 0.13
Estimated number of clusters: 3
Estimated number of noise points: 37041
Pct of noise points: 0.370410
```

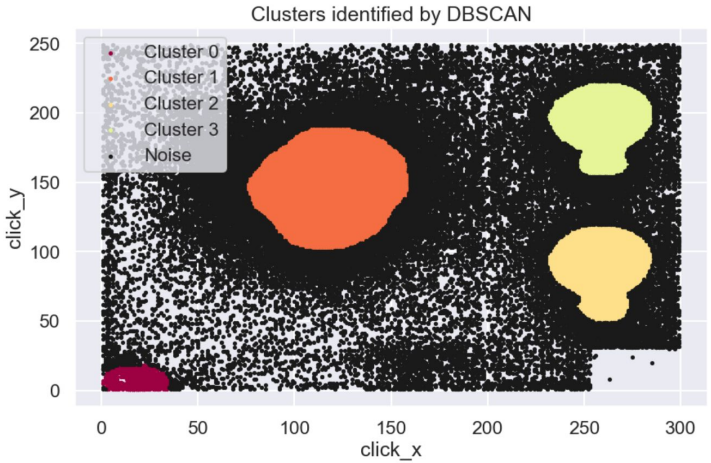


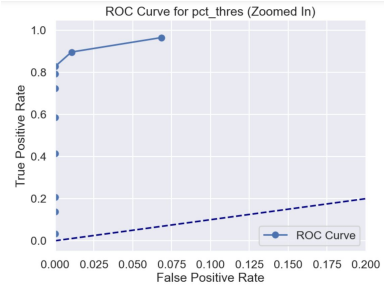

```
------------------------
Epsilon: 0.13 delta pct thres: 0.00 Pct_thres: 0.63

 Broken pct: 0.14, Total in CB: 27, Missed in CB: 1, Pct missed: 0.04, Not in CB: 2
confusion matrix (rate) ((TPR,FNR),(FPR,TNR)):
 ( 0.96 , 0.04 )
 ( 0.01 , 0.99 )
```

- **100k** bootstrapped samples to generate clusters.
- Use enhanced **5k** bootstrapped samples per domain.
  - The two grid's perform very well upon hyperparameter tuning:
    - Grid_id = 333346 performs best with eps = 0.13 and 1k min_sample

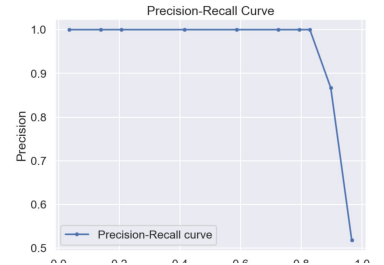
Click clustering method:

```
Epsilon: 0.13
Estimated number of clusters: 4
Estimated number of noise points: 38634
Pct of noise points: 0.521510
```






```
ROC AUC: 0.86
```




```
Epsilon: 0.15 delta pct thres: 0.00 Pct_thres: 0.57

 Broken pct: 0.06, Total in CB: 29, Missed in CB: 5, Pct missed: 0.17, Not in CB: 0
confusion matrix (rate) ((TPR,FNR),(FPR,TNR)):
 ( 0.83 , 0.17 )
 ( 0.00 , 1.00 )
```

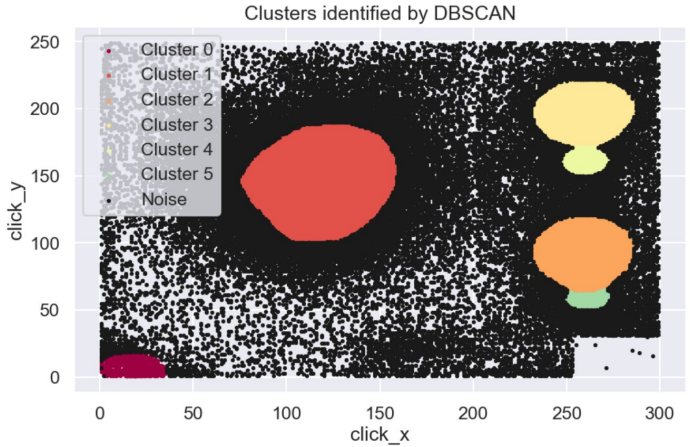
100k bootstrapped samples to generate clusters.

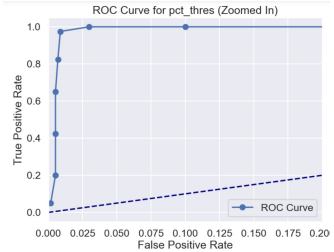
Use enhanced **5k** bootstrapped samples per domain.

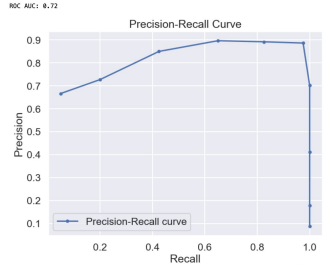
- The two grid's perform very well upon hyperparameter tuning:
  - Grid_id = 333519 performs best with eps = 0.13 and 1k min_sample
  - We remove "corner clusters".

Performance across grid:

```
Epsilon: 0.13
Estimated number of clusters: 6
Estimated number of noise points: 39161
Pct of noise points: 0.391610
```








```
Epsilon: 0.13 Pct_thres: 0.64

 Broken pct: 0.07, Total in CB: 40, Missed in CB: 1, Pct missed: 0.03, Not in CB: 5
confusion matrix (rate) ((TPR,FNR),(FPR,TNR)):
 ( 0.97 , 0.03 )
 ( 0.01 , 0.99 )
```

- Correspondingly, translating the same parameters from one to the does perform well:
  - 333346 → 333519:
    - **Remove corner cluster**:
      - **TPR**: 0.97 (1/40 missed)
      - **FNR**: 0.03 (5/600-ish)
    - Include corner cluster:
      - TPR: 0.8 (8/40 missed)
      - FNR: 0.0 (0/600-ish)
  - 333519 → 333346:
    -