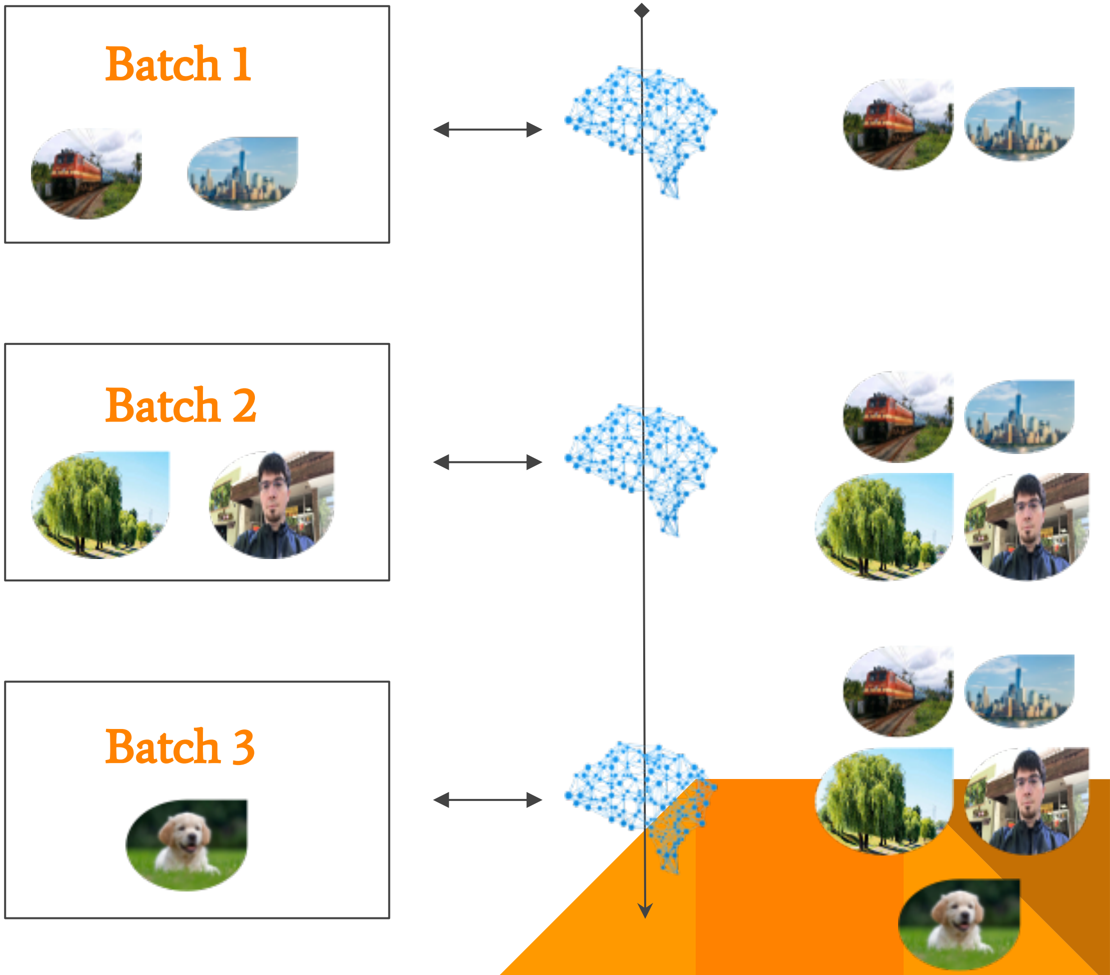
Incremental Learning

Filippo Barba   Francesco Guardamagna   Matteo Villosio

Introduction

Incremental Learning is a strategy in machine learning where data is fed to the network incrementally over time, through datastream.



Properties of an Incremental Learning Algorithm

1. Trainable from a stream of data in which examples of different classes occur at different times
2. Provide a competitive multi-class classifier for the classes observed until then
3. Memory footprint should remain bounded

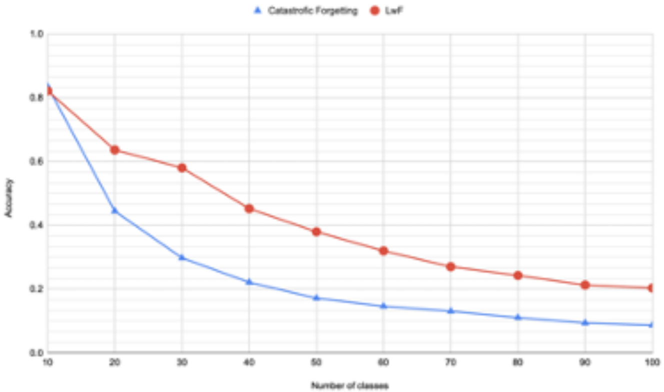
Getting to iCaRL

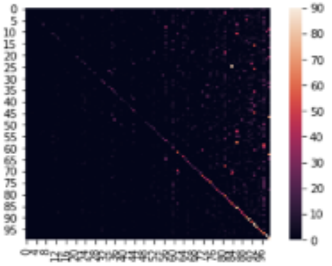
Finetuning

- Simplest method
- No strategy to avoid catastrophic forgetting
- Minimal Memory footprint
- Fast but with poor performances

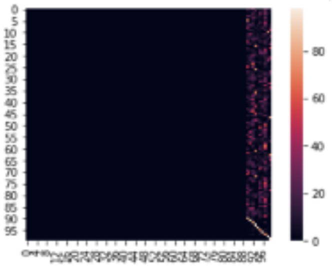
Learning without Forgetting

- Introduction of distillation loss
- No data from previous classes
- First step towards preserving previous knowledge





Finetuning



Learning without Forgetting

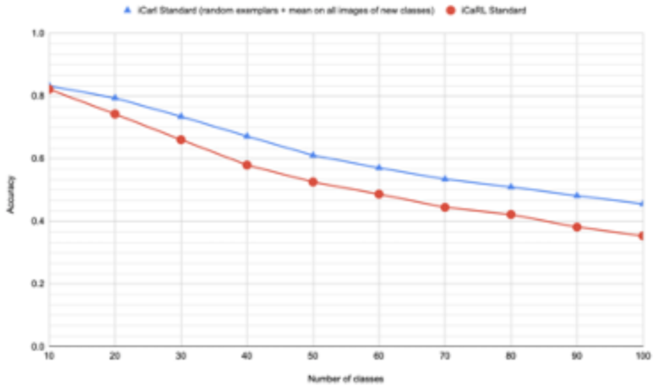
iCaRL

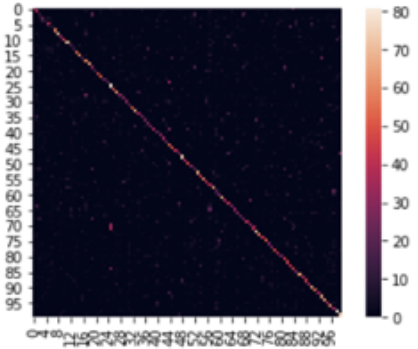
Standard iCaRL

- Exemplars to maintain examples of previous classes
- Selection of the exemplars, approximating class mean
- Distillation loss as regularization term
- Nearest mean classification on available exemplars

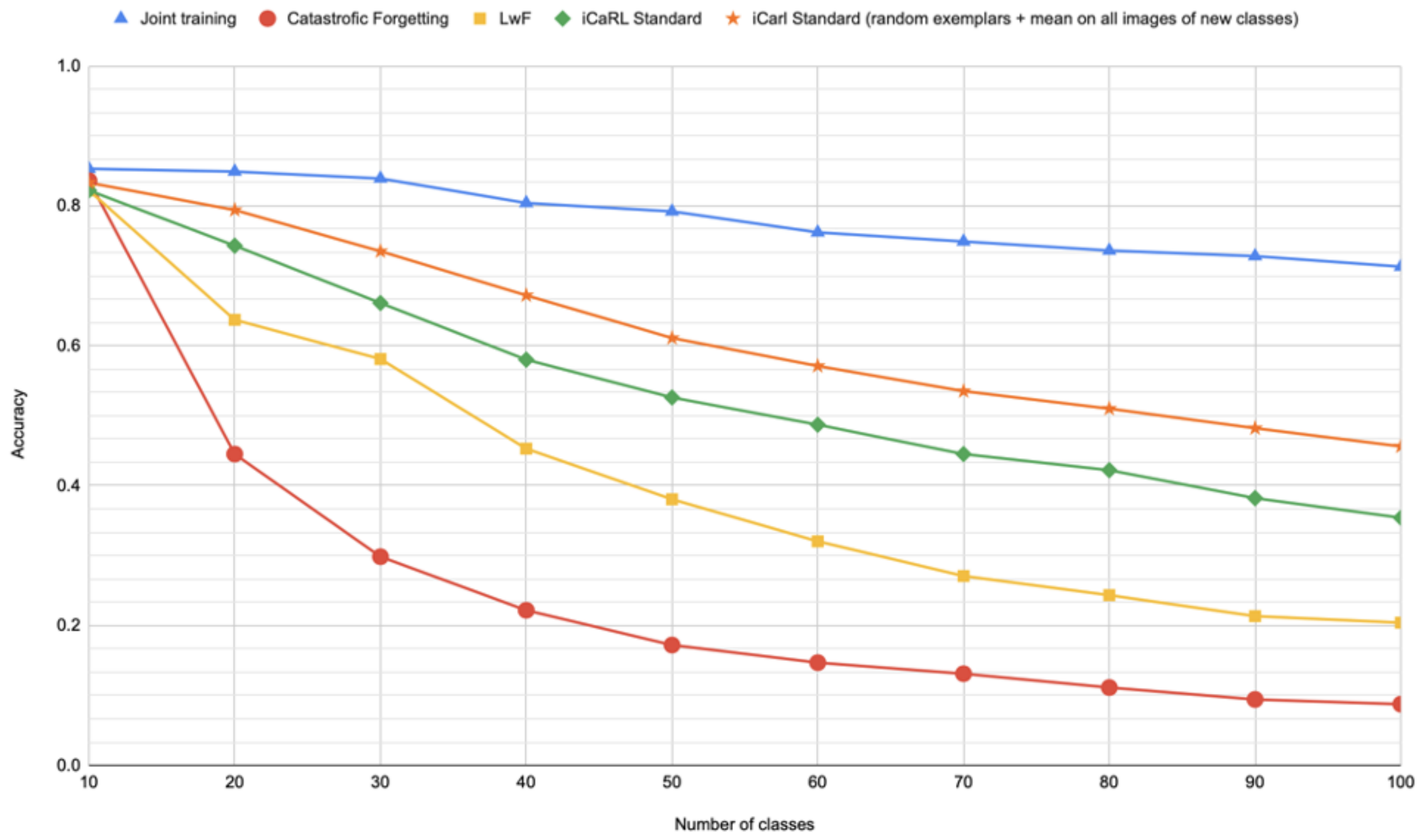
Our iCaRL

- Exemplars selected randomly
- Nearest mean classification on current batch and available exemplars





iCaRL



Joint training
Catastrofic Forgetting
LwF
iCaRL Standard
iCarl Standard (random exemplars + mean on all images of new classes)
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
10
20
30
40
50
60
70
80
90
100
Number of classes

Ablation studies:
Different approaches for losses and classifiers

Losses

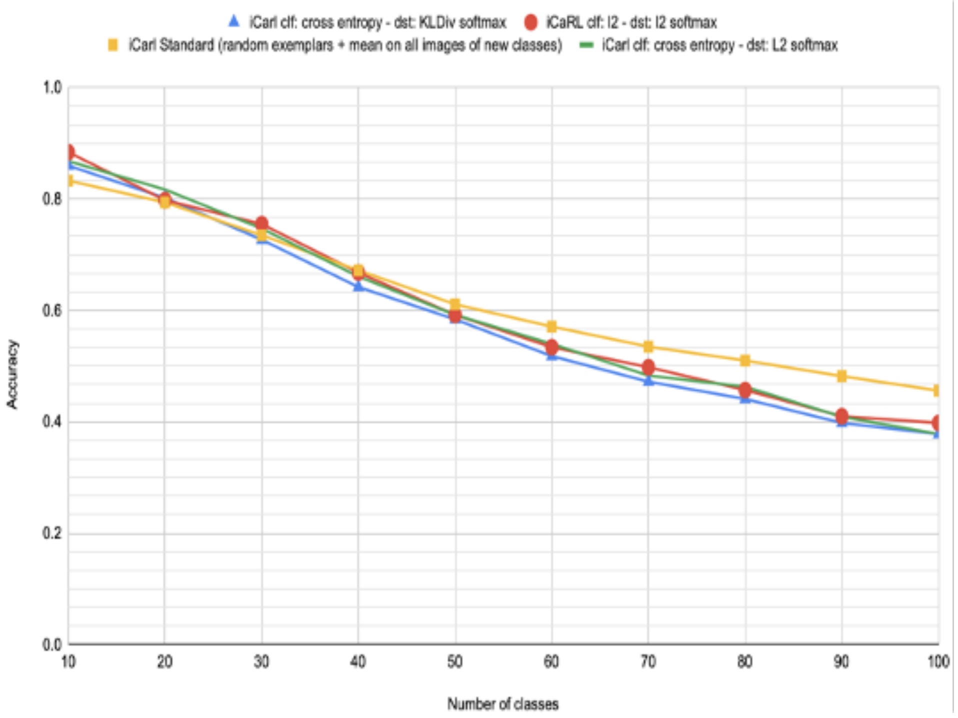
Different combinations of loss functions

iCaRL's approach:

- BCE with logits

Our approaches:

- CrossEntropy as classification and KLDivLoss as distillation
- CrossEntropy as classification and L2 as distillation
- L2 as classification and L2 as distillation



Classifiers

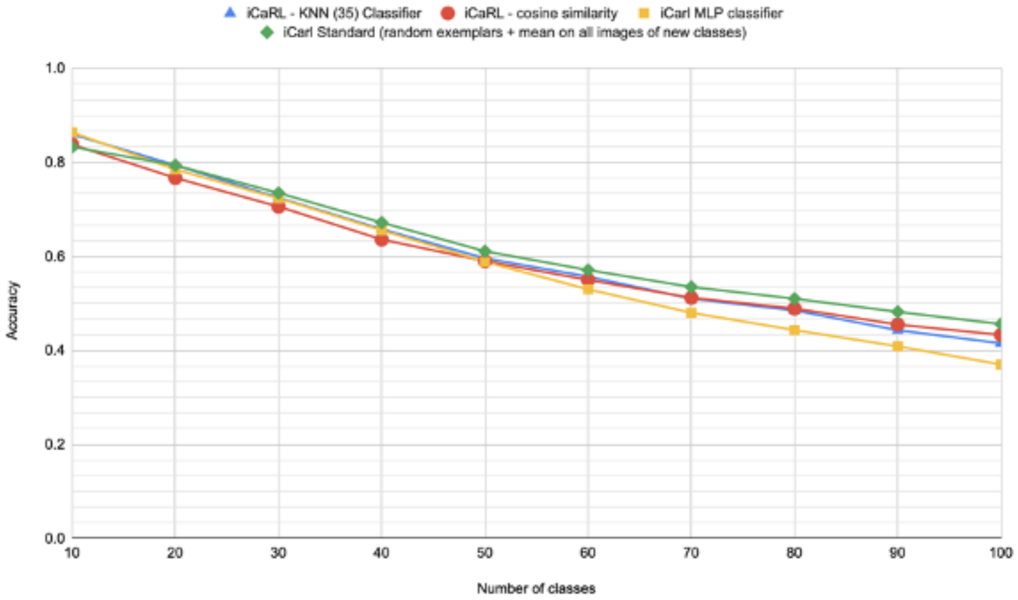
Classifiers

iCaRL's approach:

- Nearest Mean of Exemplars (NME)

Our approaches:

- K-Nearest Neighbors
- Cosine similarity
- MultiLayer Perceptron

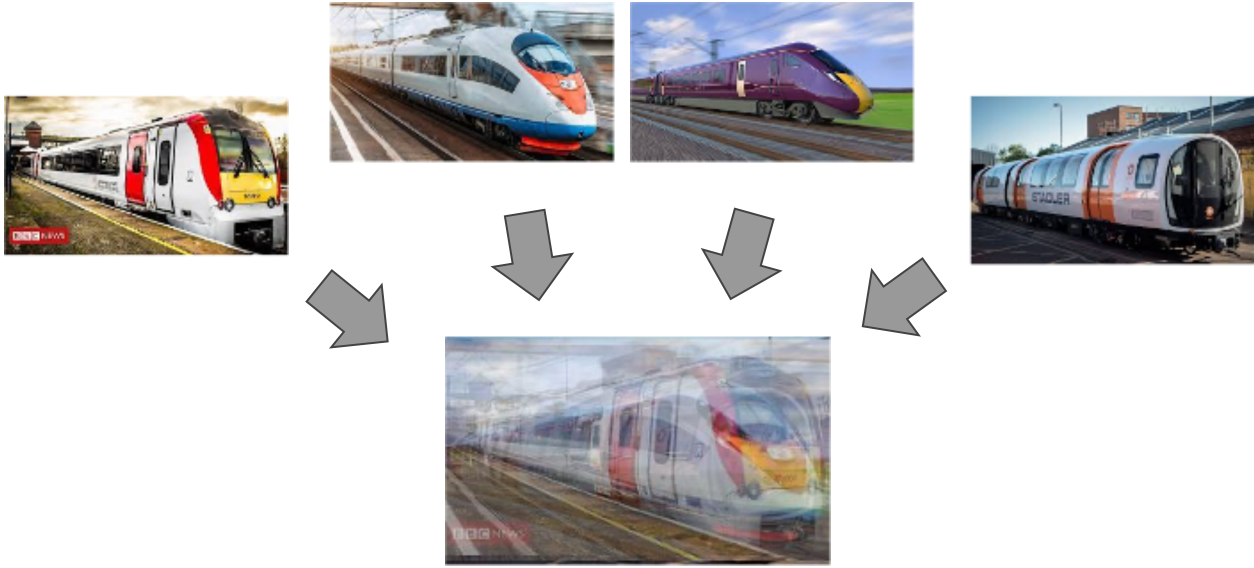


Our Approach

Introduction

The Goal

To preserve more features using the same number of exemplars by using dataset condensation.



GENERATING MEAN IMAGES WITH SAME WEIGHTS

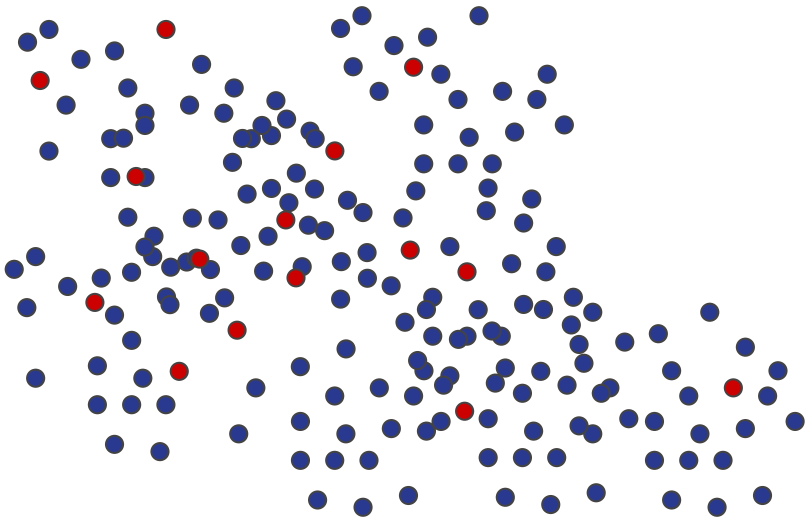
Random images with same weight

- Randomly sample N images from a class.
- Mean on randomly sampled images to generate one exemplar.

Clustered images with same weight

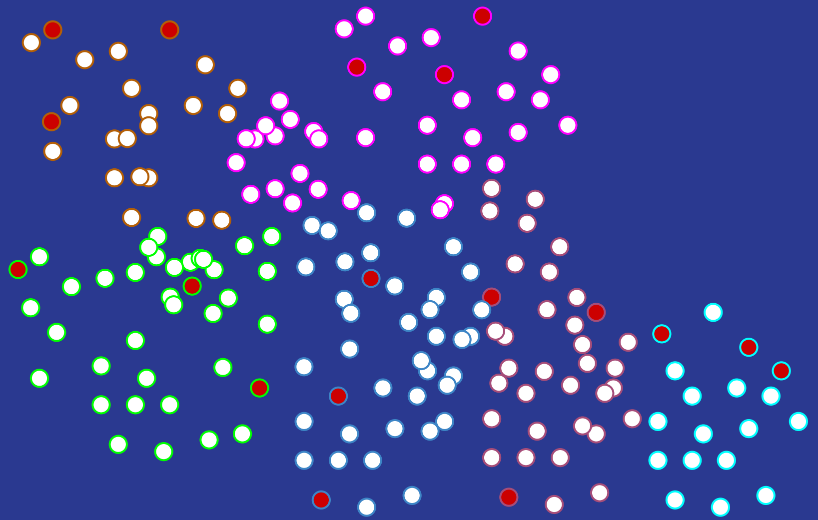
- Clusters generated on feature space using K-means.
- Images sampled from clusters.
- Mean on randomly sampled images from all clusters to generate one exemplar.

Random samples



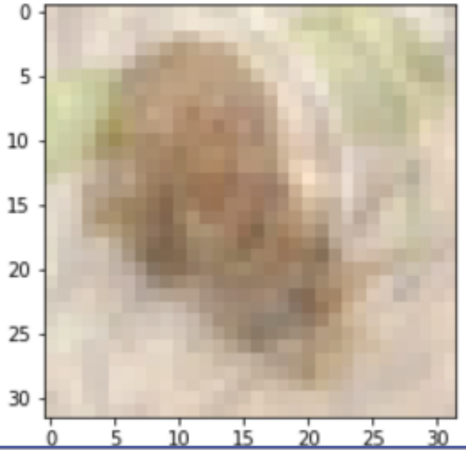
Randomly samples images

Random samples from clusters



Randomly samples images per cluster

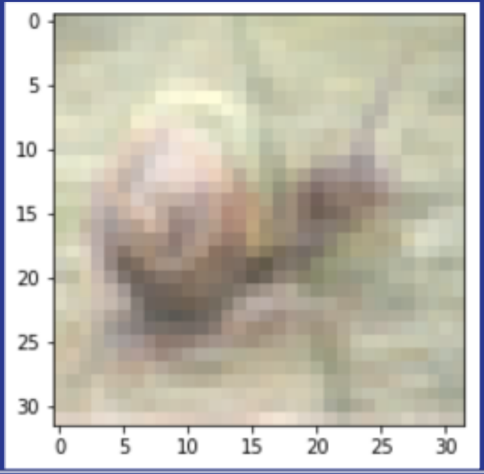
Random samples



Algorithm 1 Random Sampling Average Image Generation
CONSTRUCTEXEMPLARSETMEANIMAGESAVERAGE

input images $X = \{x_1, \ldots, x_n\}$ of class y
input m target number of average images
input r number of images to average
  for $k = 1, \ldots, m$ **do**
    $Z = \{z_1, \ldots, z_r\}$ randomly sampled images from X
    $\mu_k \leftarrow \frac{1}{r} \sum_{i=1}^{r} z_i$
  end for
  $M \leftarrow (\mu_1, \ldots, \mu_m)$
output average image set M

Random samples from clusters



Algorithm 2 Clustered images with same weight CONSTRUCTEXEMPLARSETMEANIMAGESAVERAGECLUSTER

input images $X = \{x_1, \ldots, x_n\}$ of class y
input m number of exemplars to generate
input l number of clusters
input f samples per cluster
  $C = \{c_1, \ldots, c_l\}$ clusters of images from X
  for $k = 1, \ldots, m$ **do**
    $Z \leftarrow$ collection of randomly sampled images from C clusters
    $\mu_k \leftarrow \frac{1}{\#Z} \sum_{i \in Z} z_i$
  end for
  $M \leftarrow (\mu_1, \ldots, \mu_m)$
output average image set M for class y

CLUSTERED IMAGES WITH DISTANCE BASED WEIGHTS

DISTANCE CENTROIDS-CLASS_MEAN

- Take k random images from a cluster and compute their mean.
- Add the mean to the exemplar we are generating, weighing this contribution using the distance of the centroid of the cluster to the class mean, in the feature space.
- Execute this procedure for all clusters, to generate one exemplar.

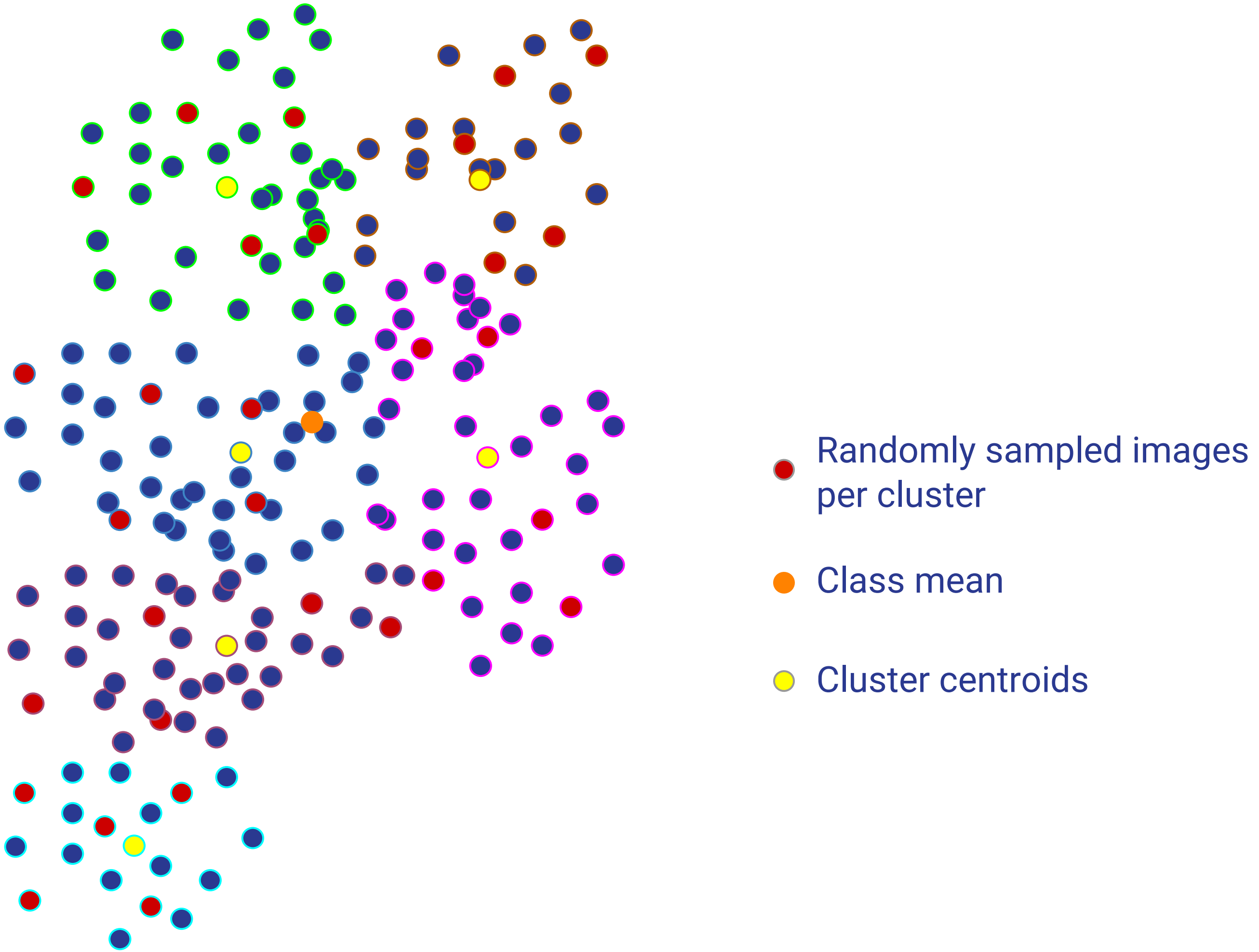
DISTANCE FROM THE CENTROID OF ONE CLUSTER

- For each exemplar, select one random cluster.
- Select k random images from that cluster.
- Add all the images to the exemplar we are generating, weighing them by their distance to the centroid of the cluster.

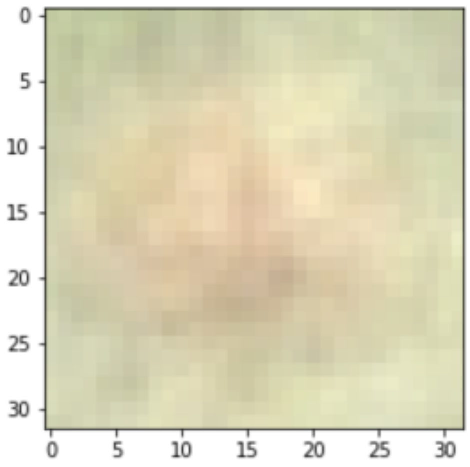
COMBINATION OF BOTH TECHNIQUES

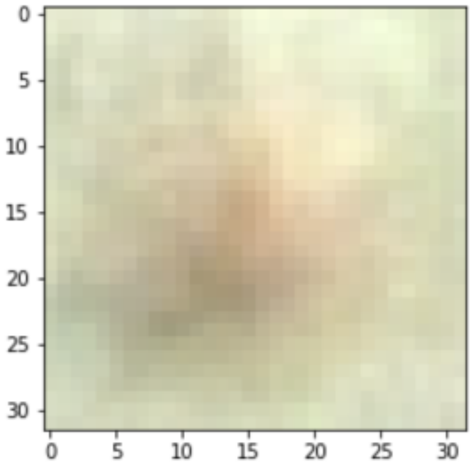
- Take k random images from a cluster and compute their mean, weighing each image with its distance to the centroid of the cluster.
- Add this contribution to the exemplar we are generating , weighing it with the distance centroid - class_mean.
- Execute this procedure for all clusters, to generate one exemplar.

Clustered images with distance based weights



Clustered images with distance based weights





Algorithm 3 Clustered images with distance-based weight ConstructExemplarSetMeanImagesClustersDistance

input images $X = \{x_1, \ldots, x_n\}$ of class y
input m number of exemplars to generate
input r number of images to average per cluster
input l number of clusters
input f samples per cluster
  $C = \{c_1, \ldots, c_l\}$ clusters from X
  for $k = 1, \ldots, m$ **do**
    $Z_j \leftarrow$ collection of randomly sampled images from cluster C_j
  $\mu_k \leftarrow \frac{1}{\sum \alpha} \sum_{j \in C} \alpha_j \frac{1}{f} \sum_{i \in C_j} \mathbf{z}_j$ with α_i weight distance of the i-th datapoint
  end for
  $M \leftarrow (\mu_1, \ldots, \mu_m)$
output average image set M for class y

GENERATING MEAN IMAGES WITH UNBALANCED WEIGHTS

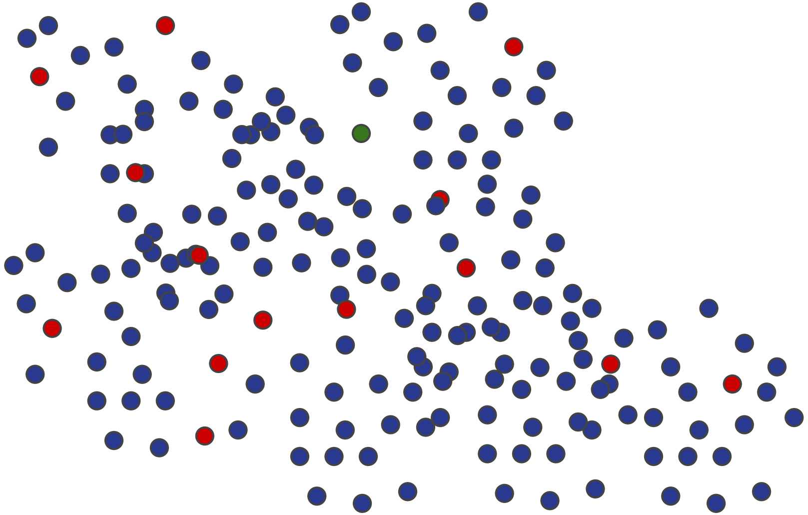
Random images with unbalanced weights

- Randomly sample N images from a class.
- Combine random images by giving one 80% contribution, and average the others over the remaining 20%.

Clustered images with unbalanced weight

- Clusters generated on feature space.
- Images sampled from clusters.
- Combine random images from a cluster by giving the image nearest to the centroid 80% contribution, and average ***N*** more images weighed by the remaining 20%.
- Separate feature extractor vs standard model

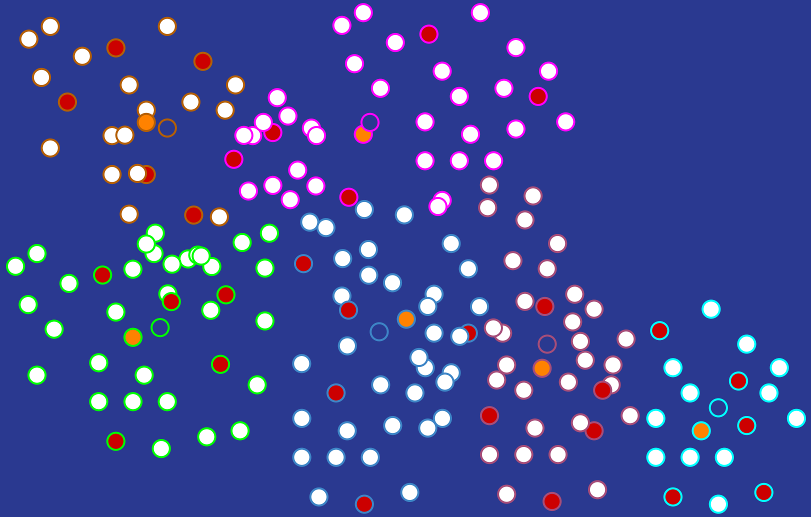
Random samples



Randomly sampled image with 80% contribution

Randomly samples images averaged to 20% contribution

Random samples from clusters

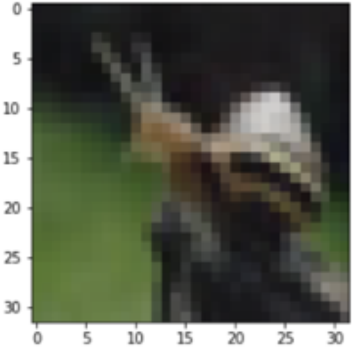


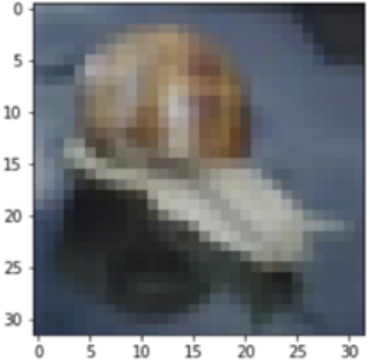
Cluster centroids

Nearest image to centroid, 80% contribution

Random images from cluster, averaged to 20% contribution

Random samples

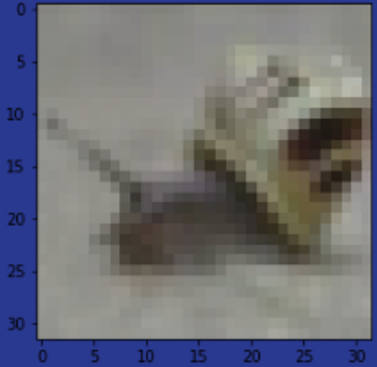




Algorithm 4 Random images with unbalanced weights CONSTRUCTEXEMPLARSETMEANIMAGESUNBALANCED

input images $X = \{x_1, \ldots, x_n\}$ of class y
input m number of exemplars to generate
input s randomly samples per exemplar
  $Z \leftarrow X$
  for $k = 1, \ldots, m$ **do**
    $C = \{c_1, \ldots, c_s\}$ sampled images from Z
    $Z \leftarrow Z \setminus \{c_1\}$
    $\mu_k \leftarrow 0.8 \cdot c_1 + 0.2 \frac{1}{s-1} \cdot \sum_{i=2}^{s} c_i$
  end for
  $M \leftarrow (\mu_1, \ldots, \mu_m)$
output average image set M

Random samples from clusters



Algorithm 5 Clustered images with unbalanced weights CONSTRUCTEXEMPLARSETMEANIMAGESCLUSTERSUNBALANCED

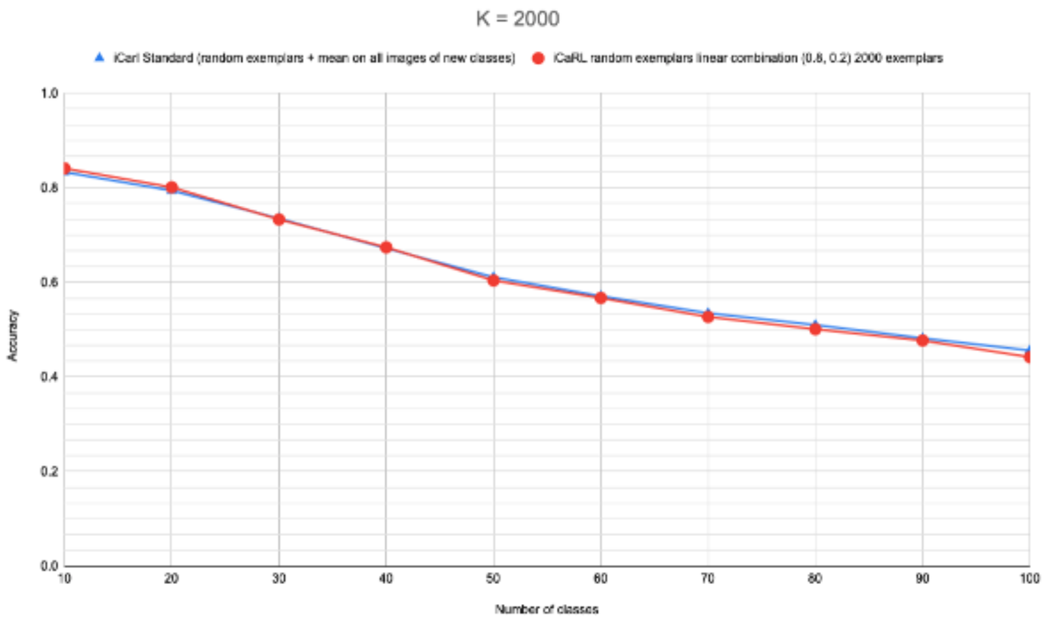
input images $X = \{x_1, \ldots, x_n\}$ of class y
input m number of exemplars to generate
input r number of images to average per cluster
input l number of clusters
input f samples per cluster
  for $k = 1, \ldots, l$ **do**
    $C = \{c_1, \ldots, c_l\}$ clusters from X
  end for
  for $k = 1, \ldots, m$ **do**
    $z_1 \leftarrow$ image closest to centroid of random cluster c_j
    $c_j \leftarrow c_j \setminus \{z_1\}$
    $\{z_2, \ldots, z_f\} \leftarrow$ randomly sampled images from c_j
    $\mu_k \leftarrow 0.8 \cdot z_1 + 0.2 \frac{1}{f-1} \cdot \sum_{i=2}^{f} z_i$
  end for
  $M \leftarrow (\mu_1, \ldots, \mu_m)$
output average image set M

Accuracy comparisons

Findings

Using the same number of exemplars as iCaRL's setting, we obtain a comparable behaviour, falling behind a few percentage points in the last batches.

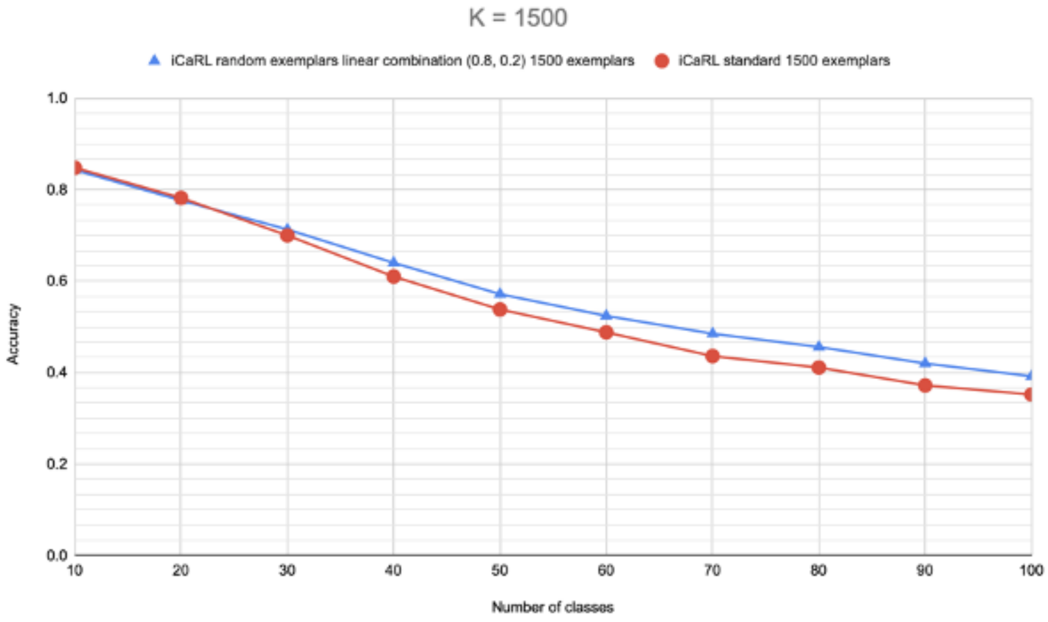
The lack of difference between the two methods could be attributed to the fact that we have enough exemplars to avoid a decrease in performance, even when introducing 100 total classes.



Accuracy comparisons

Findings

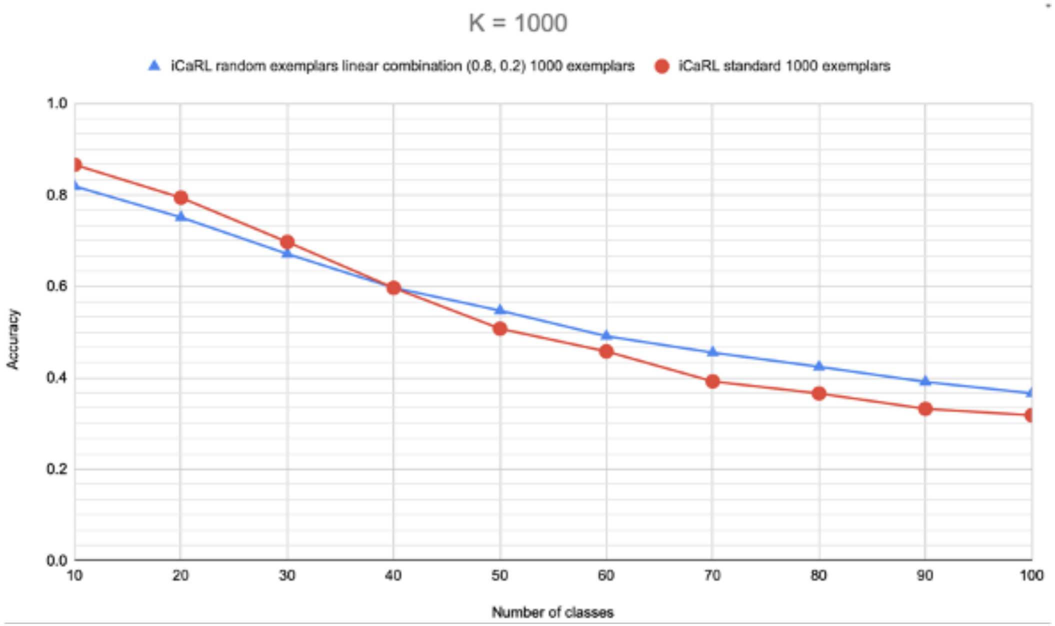
When reducing the number of total exemplars, on the other hand, our strategy preserved better knowledge of previous classes, by using more descriptive exemplars.



Accuracy comparisons

Findings

When reducing the number of total exemplars, on the other hand, our strategy preserved better knowledge of previous classes, by using more descriptive exemplars.

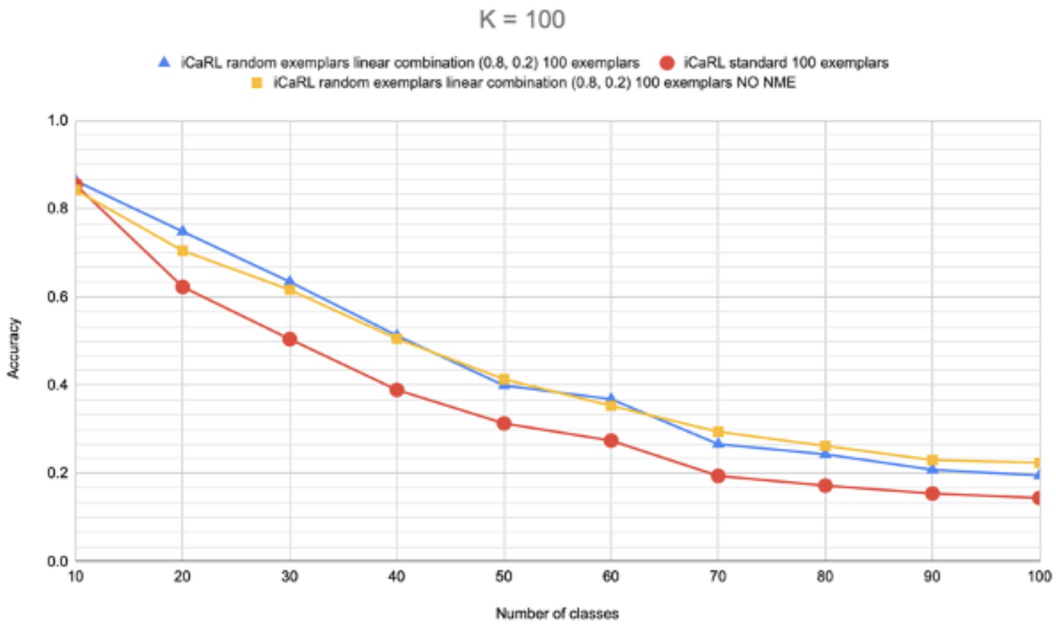


Accuracy comparisons

Findings

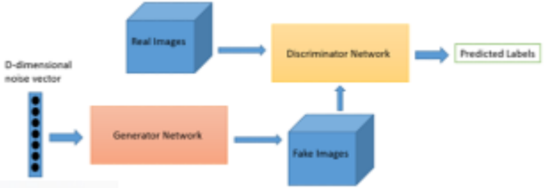
When reducing the number of total exemplars, on the other hand, our strategy preserved better knowledge of previous classes, by using more descriptive exemplars.

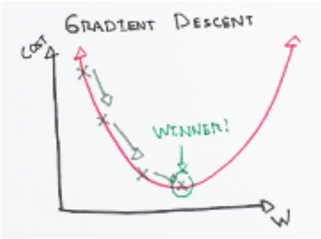
In the extreme case of using K = 100 exemplars, leaving us with a single exemplars from the 5th batch onwards, we notice an improvement with respect to standard iCaRL.



Future improvements

- **GAN Approach**
- **Gradient based Dataset Condensation**
- **More complex clustering algorithms**





Thank you

For your attention