DeepRob

Lecture 1
Image Classification
University of Michigan | Department of Robotics

DeepRob

Welcome!

Candy

Candy

Candy

Table

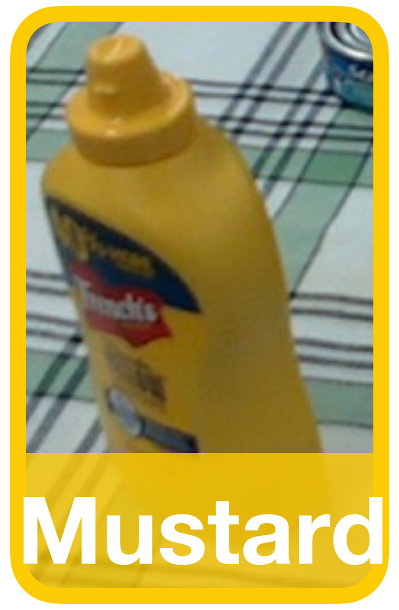
Staircase

Robot

Robot

Coffee

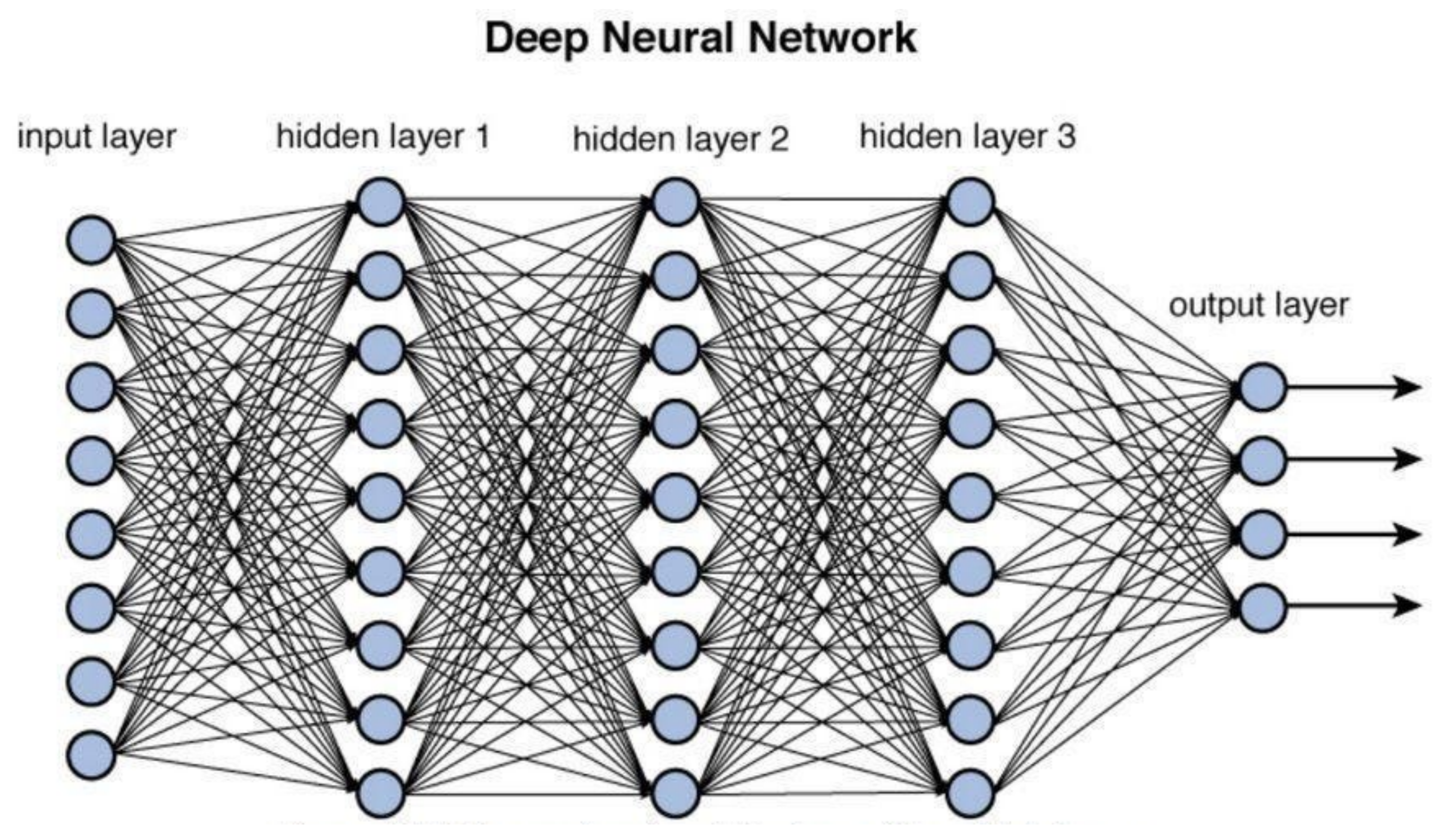
Crackers

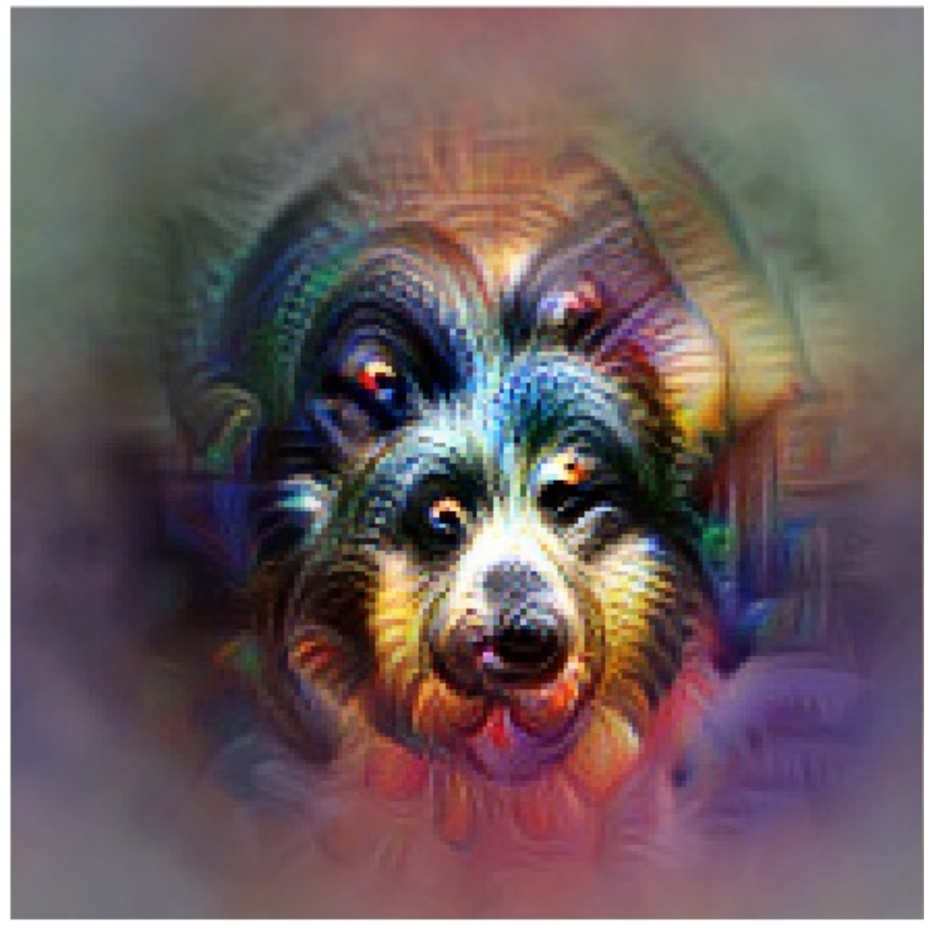


Mustard

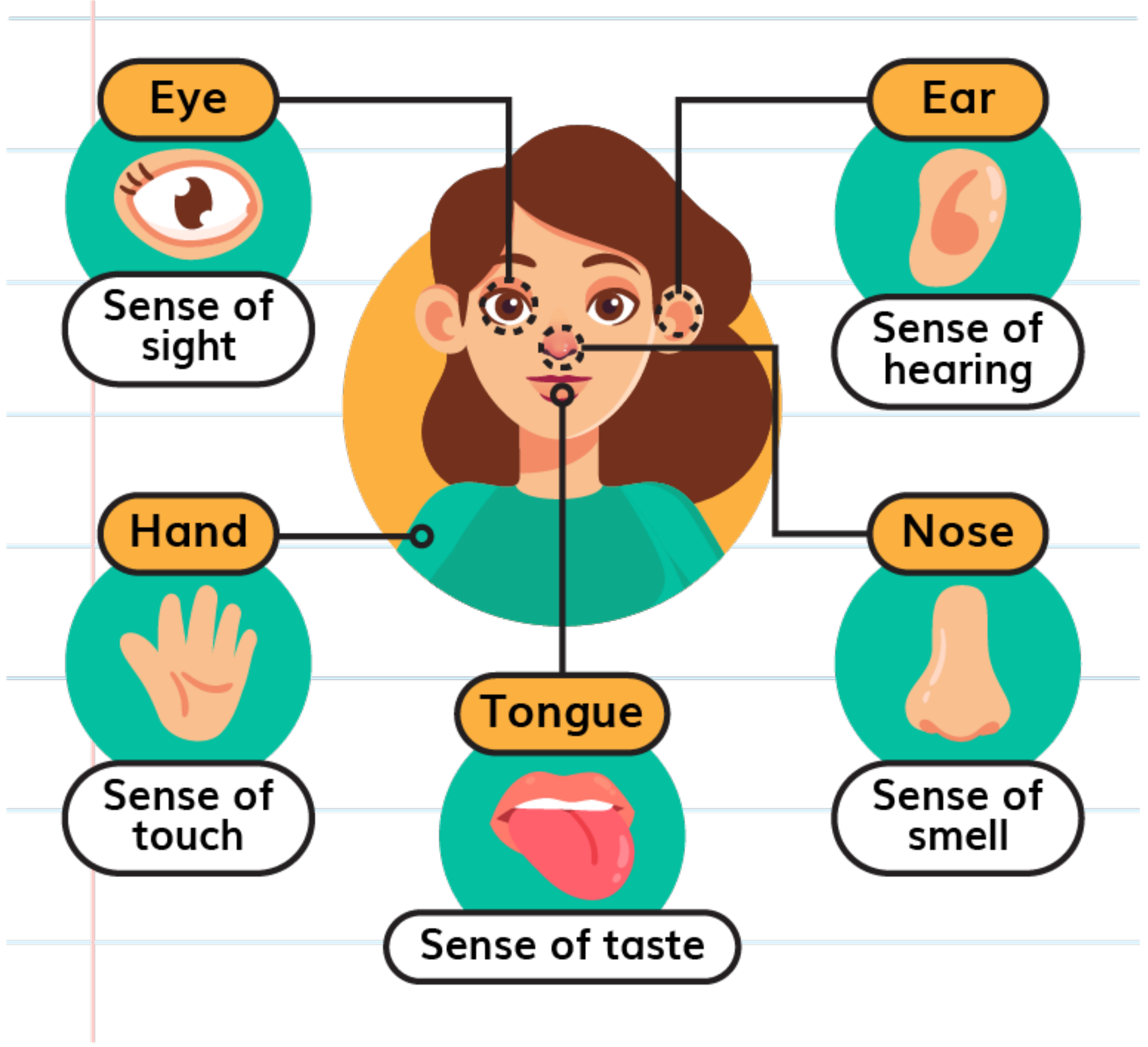
Cup

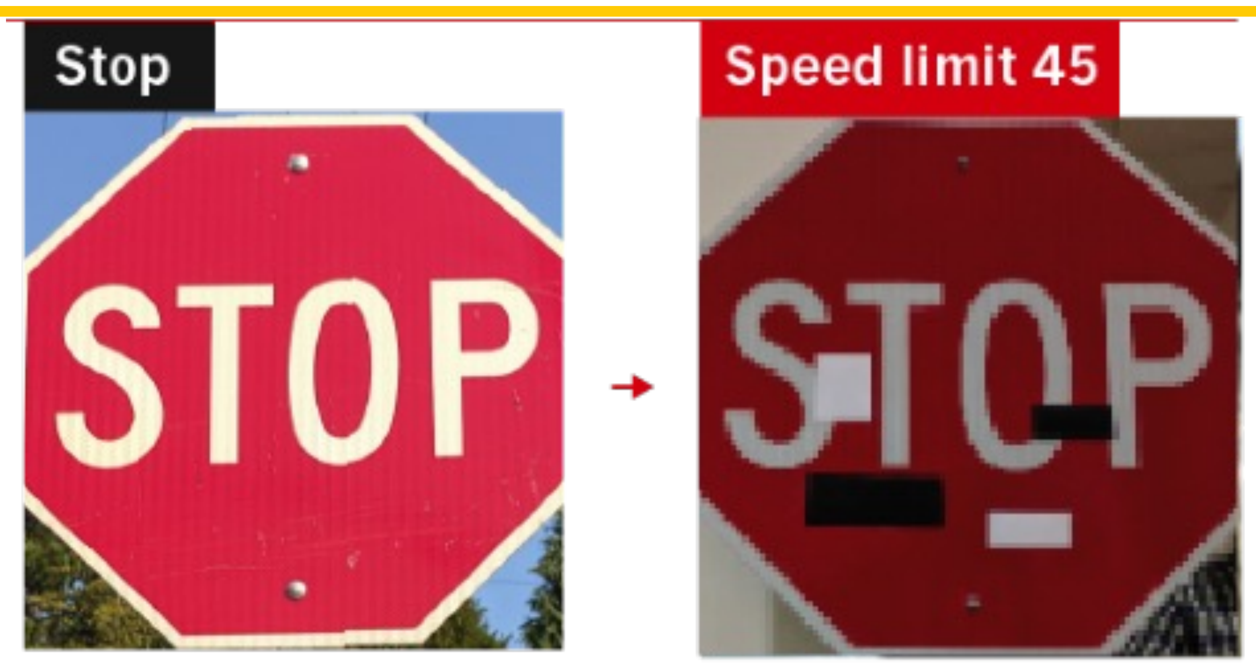
Recap: Deep Learning for Robot Perception

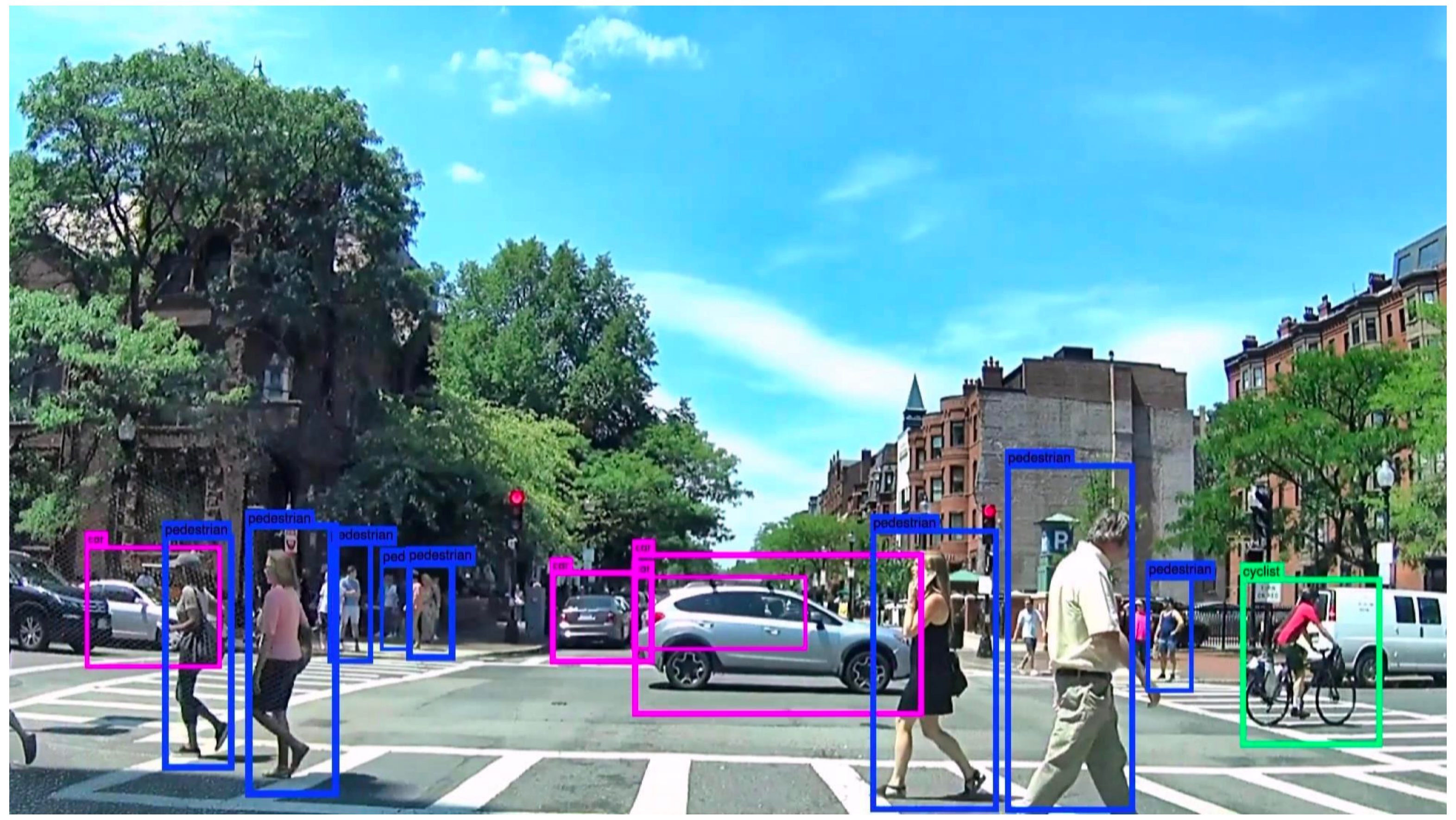


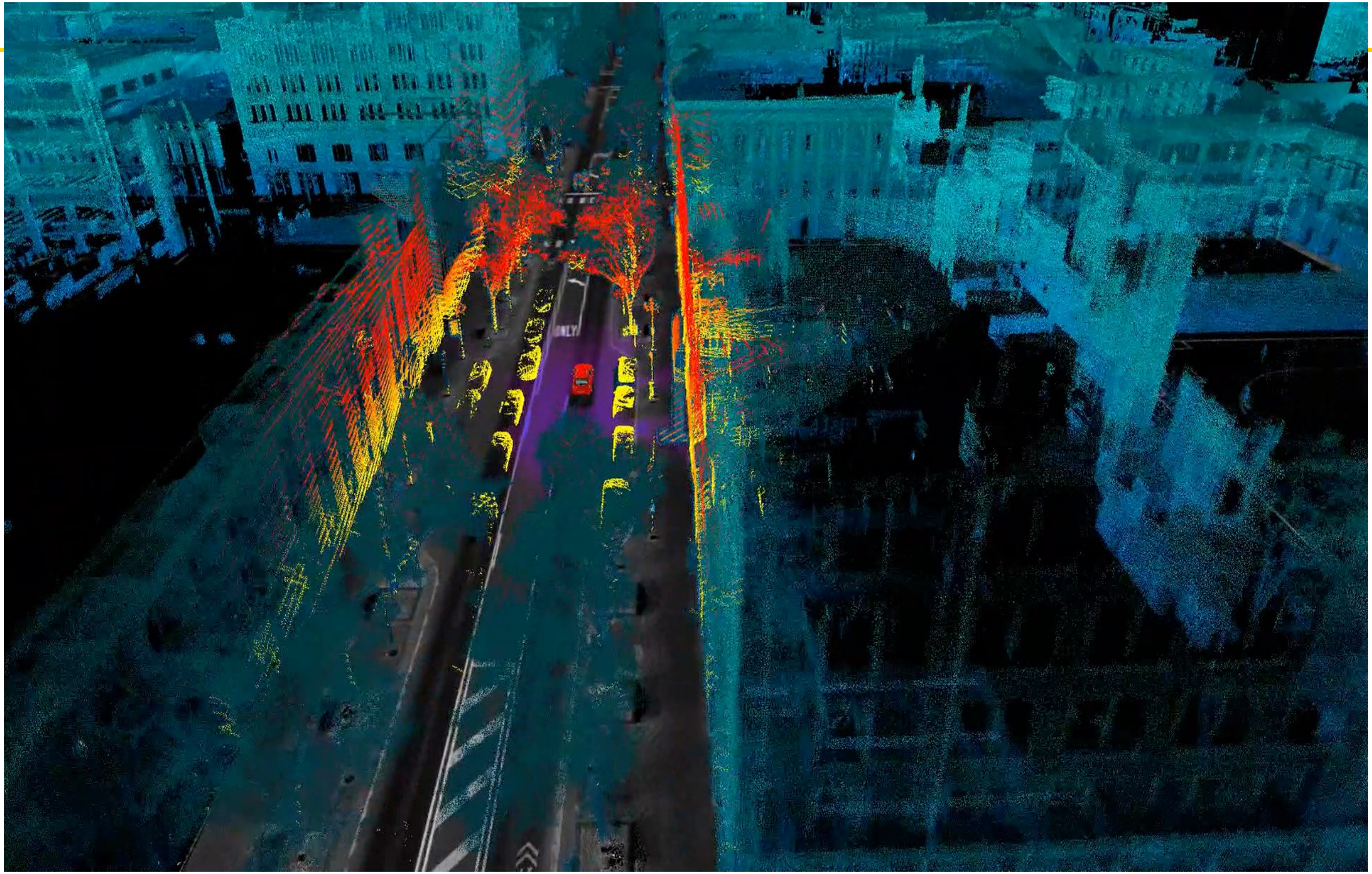


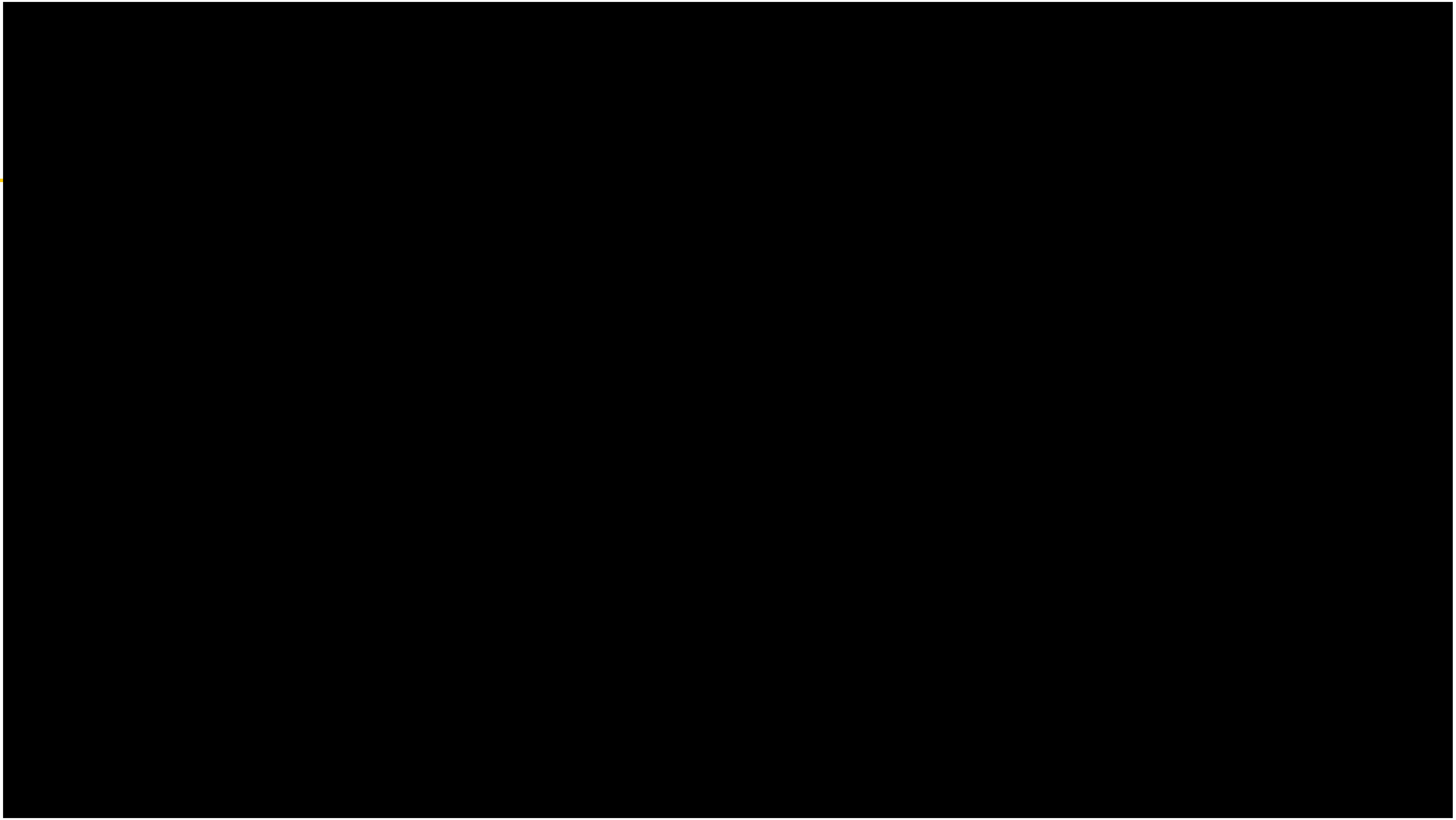
Animal faces—or snouts?

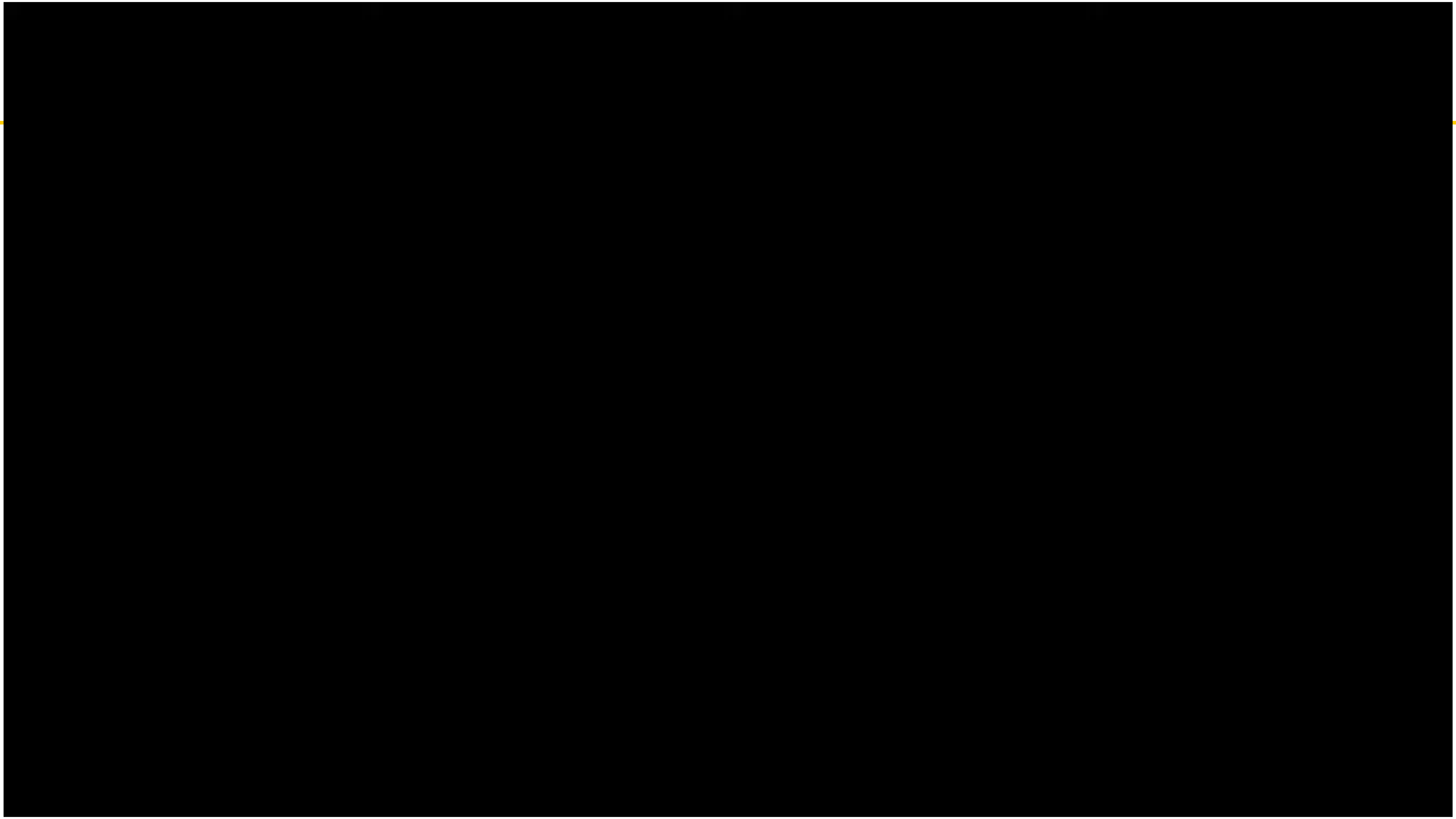







ONLY






Eigen Café


M | ROBOTICS


SNACK TIME


SNACK TIME
SKINNY POP
Chex mix
Doritos
Lay's
m&m's
HERSHEY'S

“Semantic Gap”

Input: image


```
[[183, 187, 189, 189, 188, 188, 189, 190, 186, 185, 189, 190, 187, 186, 183],
 [185, 188, 189, 188, 188, 189, 191, 193, 187, 190, 191, 189, 186, 185, 185],
 [186, 189, 189, 187, 187, 188, 189, 189, 192, 194, 189, 184, 182, 185, 187],
 [188, 188, 188, 190, 190, 189, 189, 190, 190, 189, 185, 184, 185, 188, 188],
 [187, 187, 188, 192, 191, 189, 191, 193, 191, 186, 185, 189, 187, 187, 185],
 [186, 186, 189, 191, 190, 189, 190, 192, 191, 188, 190, 193, 186, 186, 184],
 [189, 186, 189, 192, 192, 190, 191, 193, 184, 188, 190, 192, 186, 187, 186],
 [191, 189, 189, 190, 189, 190, 190, 190, 183, 187, 186, 188, 187, 189, 188],
 [192, 194, 193, 189, 188, 193, 194, 191, 191, 192, 186, 186, 187, 186, 187],
 [190, 192, 193, 191, 191, 195, 194, 191, 191, 192, 188, 189, 189, 186, 188],
 [189, 188, 190, 189, 190, 189, 187, 187, 185, 190, 188, 189, 192, 192, 191],
 [191, 188, 187, 186, 188, 190, 189, 190, 186, 193, 190, 187, 194, 194, 192],
 [194, 193, 189, 186, 189, 190, 191, 194, 192, 191, 192, 194, 194, 194, 188],
 [196, 196, 196, 193, 191, 190, 191, 195, 194, 191, 193, 194, 192, 190, 187],
 [194, 193, 194, 191, 188, 189, 190, 193, 193, 191, 193, 192, 190, 190, 190],
 [197, 194, 193, 191, 188, 189, 191, 192, 192, 192, 194, 192, 190, 193, 193],
 [202, 201, 202, 200, 196, 193, 192, 192, 190, 191, 194, 193, 191, 193, 193],
 [205, 206, 207, 206, 202, 198, 196, 194, 189, 190, 191, 192, 191, 191, 190],
 [207, 207, 204, 202, 199, 198, 199, 199, 195, 192, 192, 194, 193, 191, 190],
 [205, 203, 200, 200, 199, 196, 198, 202, 199, 194, 193, 195, 193, 191, 192],
 [199, 196, 196, 201, 205, 204, 202, 202, 199, 194, 192, 193, 191, 189, 192],
 [195, 194, 193, 196, 201, 205, 205, 203, 200, 196, 195, 195, 192, 190, 192],
 [194, 194, 193, 194, 196, 199, 202, 204, 201, 200, 200, 199, 196, 195, 196],
 [194, 193, 192, 195, 197, 199, 202, 204, 200, 203, 204, 202, 199, 200, 200],
 [199, 201, 201, 200, 200, 201, 201, 205, 202, 206, 207, 205, 203, 205, 203]]
```

What the computer sees

An image is just a grid of numbers between [0, 255]

An Image Classifier

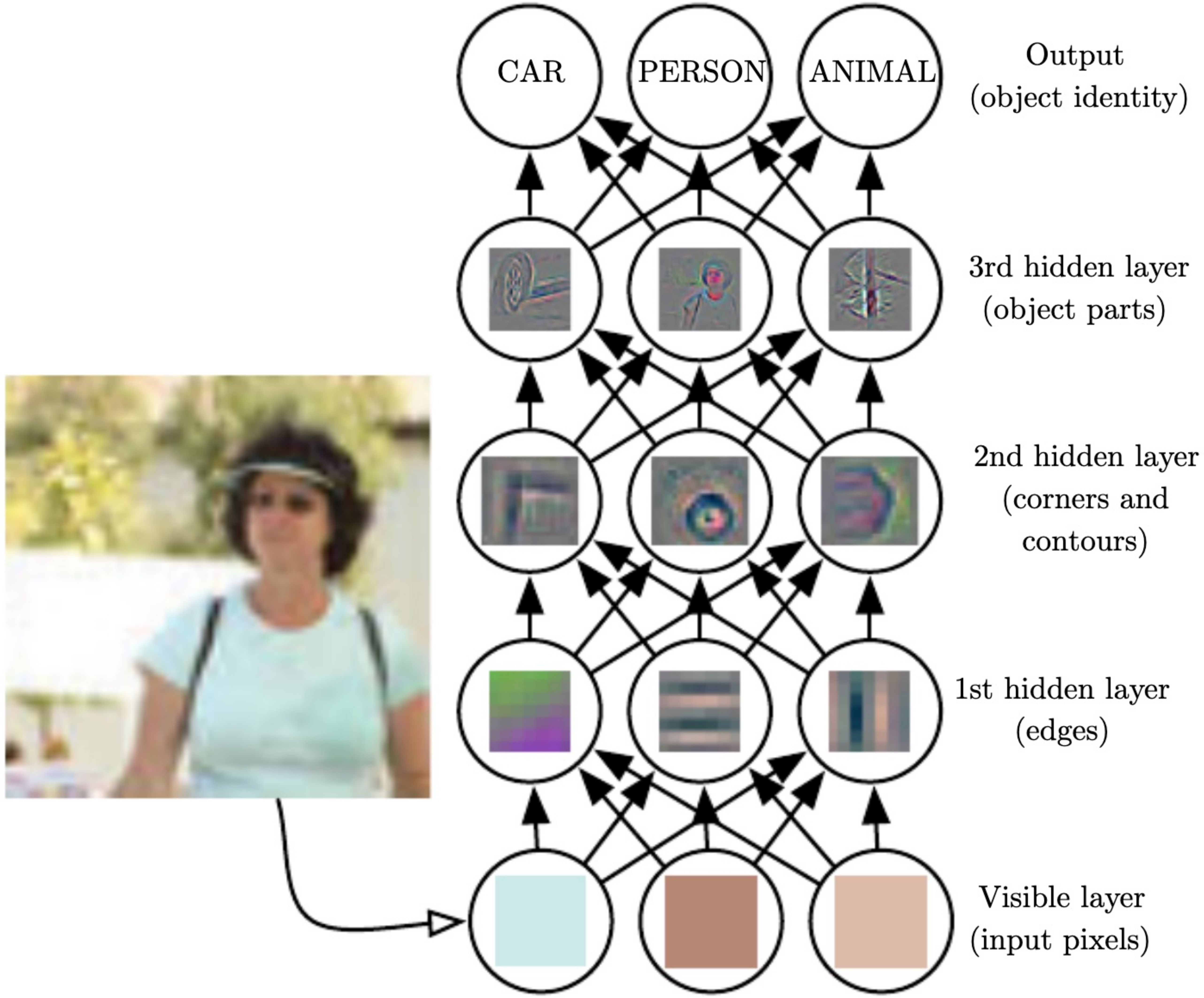
```python
def classify_image(image):
  # Some magic here?
  return class_label
```

Unlike well defined programming (e.g. sorting a list)

No obvious way to hard-code the algorithm
for recognizing each class

Deep Learning – Representation Learning

- "Features"



Deep Learning - A Data-Driven Approach

1. Collect a dataset of images + labels
2. Use ML/DL to train a classifier
3. Evaluate the classifier on new images

```
def train(images, labels):
  # Machine learning!
  return model
```

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Image Classification Datasets—CIFAR10

10 classes

32x32 RGB images

50k training images (5k per class)

10k test images (1k per class)

Alex Krizhevsky, "Learning Multiple Layers of Features from Tiny Images", Technical Report, 2009.

Image Classification Datasets—CIFAR100

100 classes
32x32 RGB images
50k training images (500 per class)
10k test images (100 per class)

20 superclasses with 5 classes each:

Aquatic mammals: beaver, dolphin, otter, seal, whale

Trees: maple, oak, palm, pine, willow

Image Classification Datasets—MNIST

Handwritten digits

10 classes: Digits 0 to 9
28x28 grayscale images
50k training images
10k test images

Due to relatively small size, results on MNIST often do not hold on more complex datasets

Image Classification Datasets—ImageNet

Deng et al., "ImageNet: A Large-Scale Hierarchical Image Database", CVPR, 2009.

Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge", IJCV, 2015.

1000 classes

~1.3M training images (~1.3K per class)
50k validation images (50 per class)
100K test images (100 per class)

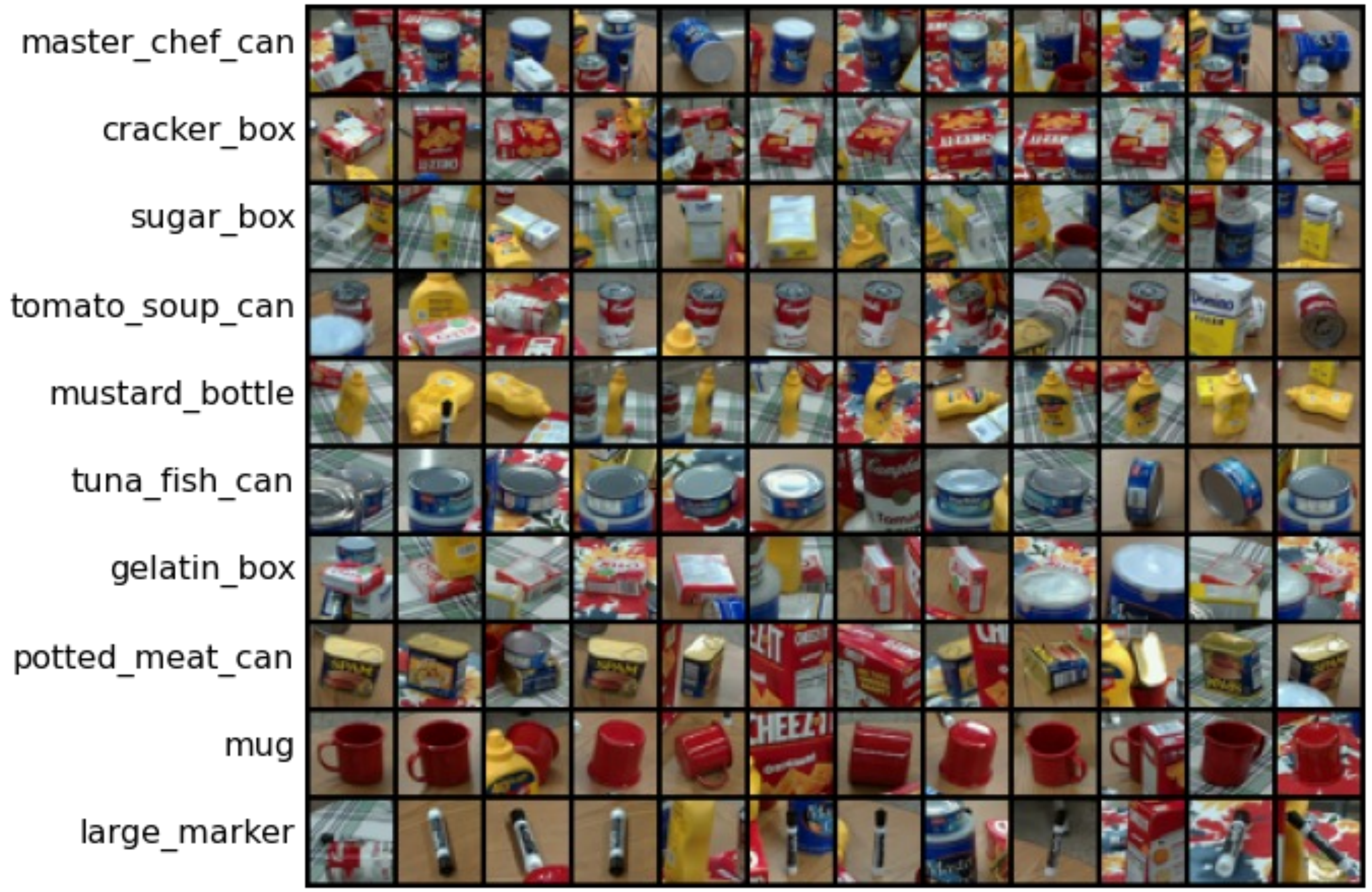
Images have variable size, but often resized to **256x256** for training

*Performance metric: **Top 5 accuracy**

Algorithm predicts 5 labels for each image, one must be right

Image Classification Datasets—PROPS

Progress **R**obot **O**bject **P**erception **S**amples **D**ataset



Chen et al., "ProgressLabeller: Visual Data Stream Annotation for Training Object-Centric 3D Perception", IROS, 2022.

10 classes
32x32 RGB images
50k training images (5k per class)
10k test images (1k per class)

Image Classification Datasets—MIT Places

Zhou et al., "Places: A 10 million Image Database for Scene Recognition", TPAMI, 2017.

365 classes of different scene types

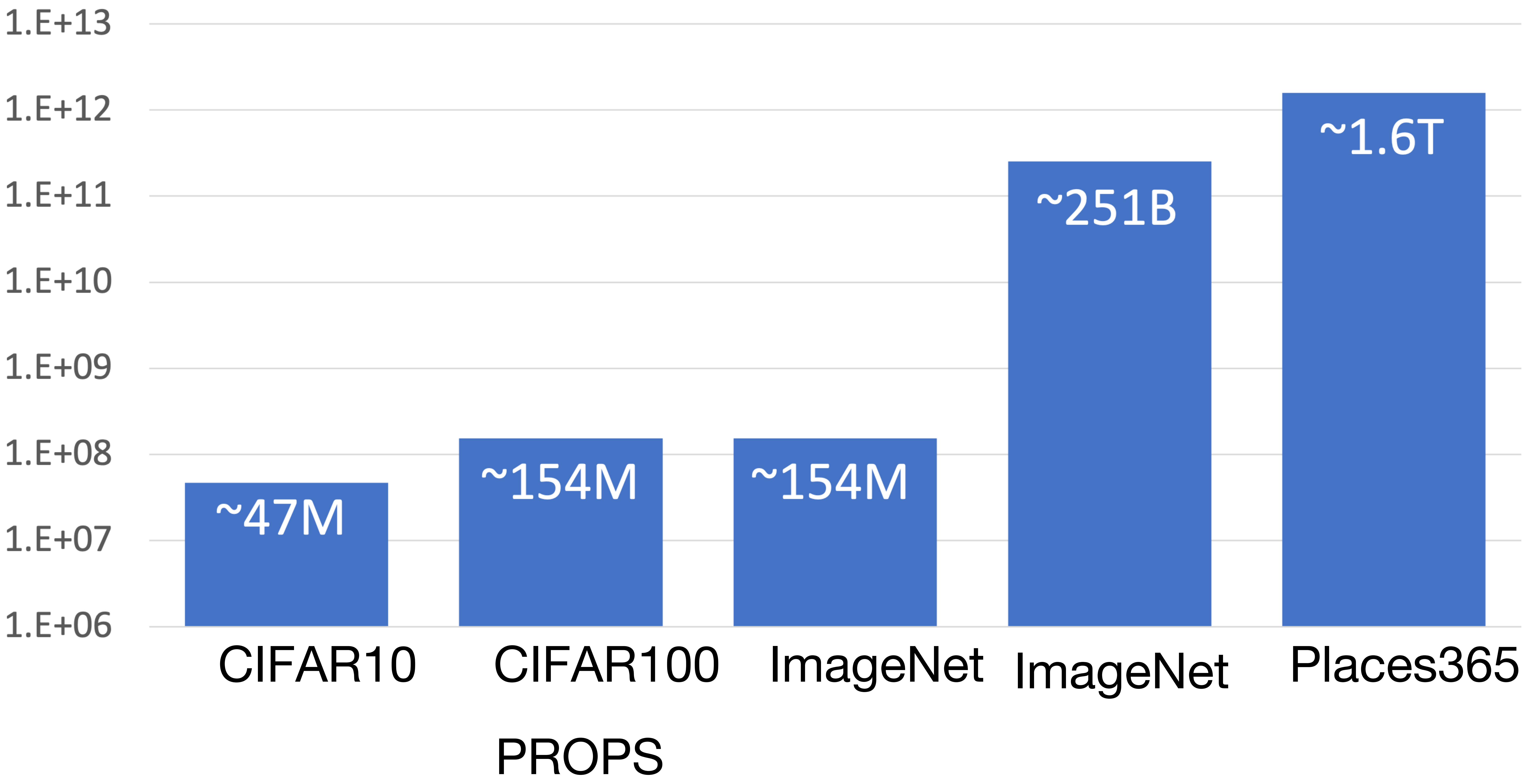
~8M training images

18.25K val images (50 per class)

328.5K test images (900 per class)

Images have variable size, but often resized to **256x256** for training

Classification Datasets—Number of Training Pixels



First Classifier—Nearest Neighbor

```
def train(images, labels):
  # Machine learning!
  return model
```

Memorize all data and labels

```
def predict(model, test_images):
  # Use model to predict labels
  return test_labels
```

Predict the label of the most similar training image

What does this look like?

Class: Mug

How do we tell the “similarity”

- Similarity/dissimilarity metrics
  - Euclidean distance: $d_E = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T (\mathbf{x}_1 - \mathbf{x}_2)}$
  - City block distance: $d_C = \sum_{i=1}^{d} |x_{1i} - x_{2i}|$
  - Mahalanobis distance: $(\mathbf{x}_1 - \mathbf{x}_2)^T \Sigma^{-1} (\mathbf{x}_1 - \mathbf{x}_2)$
  - Geodesic distance
  - Cosine angle similarity: $\cos\theta = \frac{\mathbf{x}_1^T \mathbf{x}_2}{|\mathbf{x}_1|_2^2 |\mathbf{x}_2|_2^2}$
  - and many more...

Distance Metric to Compare Images

Example: L1 distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

test image — training image — pixel-wise absolute value differences

$$\left| \begin{bmatrix} 56 & 32 & 10 & 18 \\ 90 & 23 & 128 & 133 \\ 24 & 26 & 178 & 200 \\ 2 & 0 & 255 & 220 \end{bmatrix} - \begin{bmatrix} 10 & 20 & 24 & 17 \\ 8 & 10 & 89 & 100 \\ 12 & 16 & 178 & 170 \\ 4 & 32 & 233 & 112 \end{bmatrix} \right| = \begin{bmatrix} 46 & 12 & 14 & 1 \\ 82 & 13 & 39 & 33 \\ 12 & 10 & 0 & 30 \\ 2 & 32 & 22 & 108 \end{bmatrix} \xrightarrow{\text{add}} 456$$

Nearest Neighbor Classifier

```
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Nearest Neighbor Classifier

```
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Memorize training data

Nearest Neighbor Classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

For each test image:

Find nearest training image

Return label of nearest image

Nearest Neighbor Classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Q: With N examples how fast is training?

A: O(1)

Nearest Neighbor Classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Q: With N examples how fast is training?

A: O(1)

Q: With N examples how fast is testing?

A: O(N)

Nearest Neighbor Classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

Q: With N examples how fast is training?

A: O(1)

Q: With N examples how fast is testing?

A: O(N)

This is a problem: we can train slow offline but need fast testing!

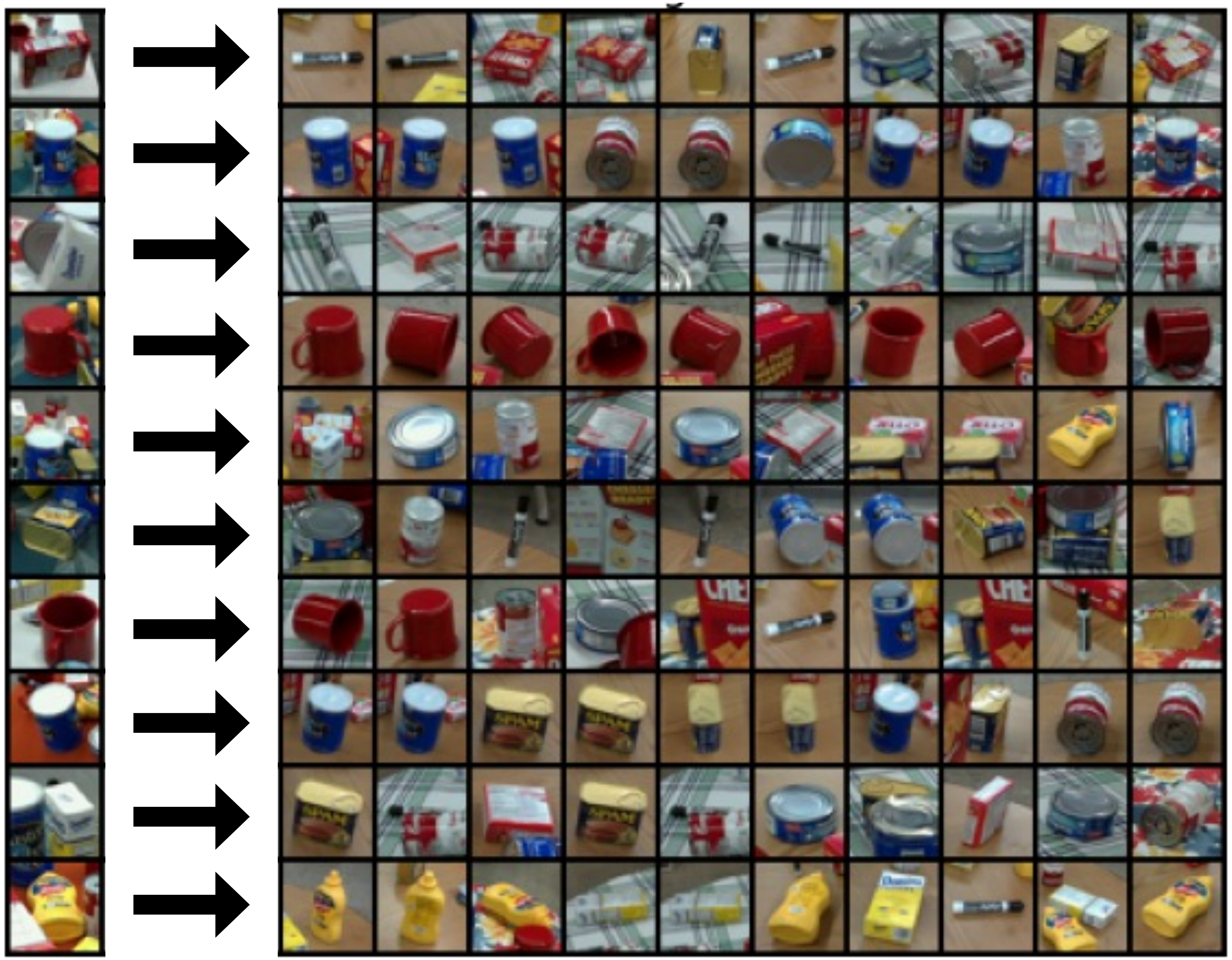
Nearest Neighbor Classifier

```python
import numpy as np

class NearestNeighbor:
  def __init__(self):
    pass

  def train(self, X, y):
    """ X is N x D where each row is an example. Y is 1-dimension of size N """
    # the nearest neighbor classifier simply remembers all the training data
    self.Xtr = X
    self.ytr = y

  def predict(self, X):
    """ X is N x D where each row is an example we wish to predict label for """
    num_test = X.shape[0]
    # lets make sure that the output type matches the input type
    Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

    # loop over all test rows
    for i in xrange(num_test):
      # find the nearest training image to the i'th test image
      # using the L1 distance (sum of absolute value differences)
      distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
      min_index = np.argmin(distances) # get the index with smallest distance
      Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

    return Ypred
```

There are many methods for fast / approximate nearest neighbors

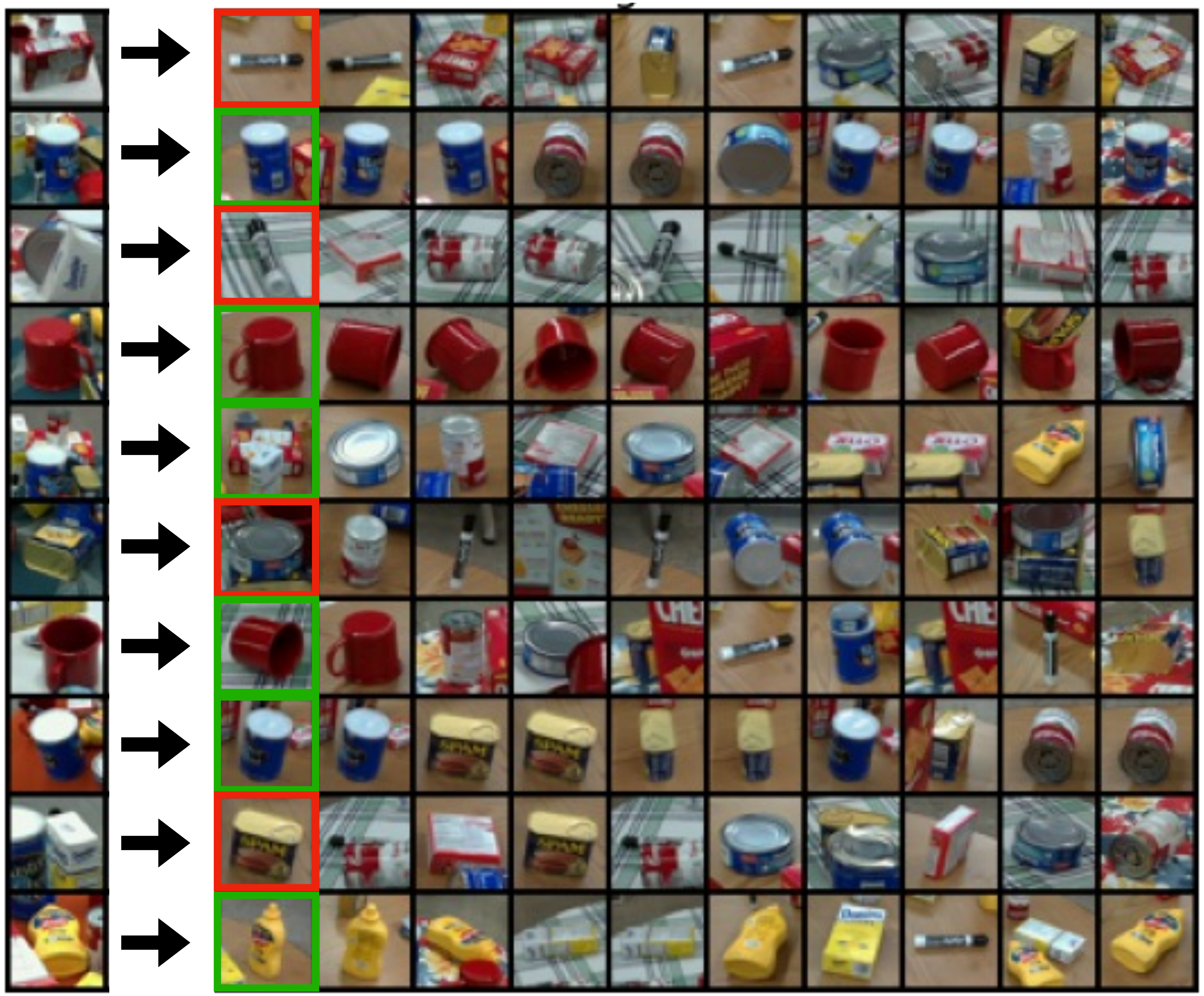
e.g. github.com/facebookresearch/faiss

What does this look like?

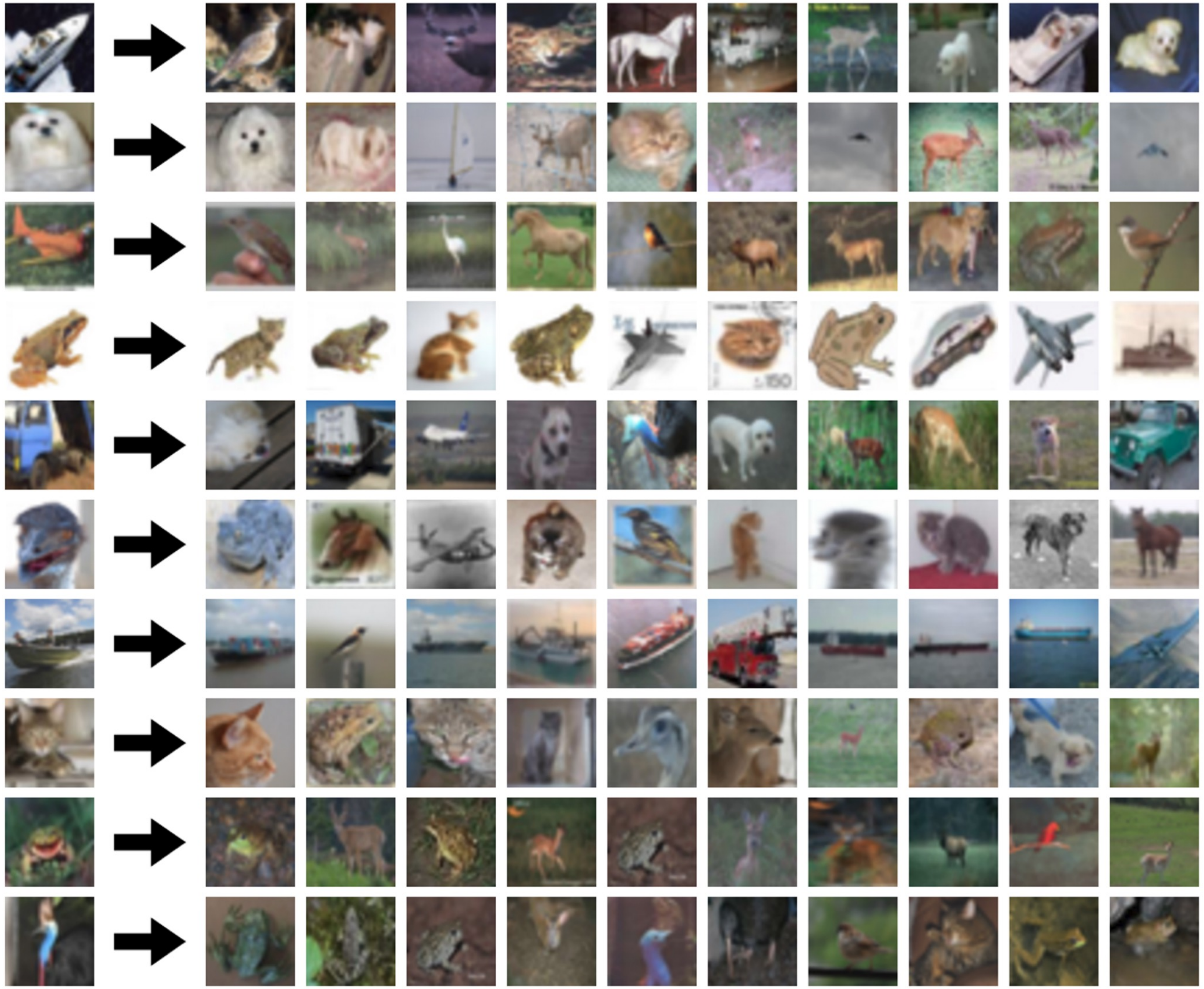


What does this look like?

PROPS dataset is instance-level



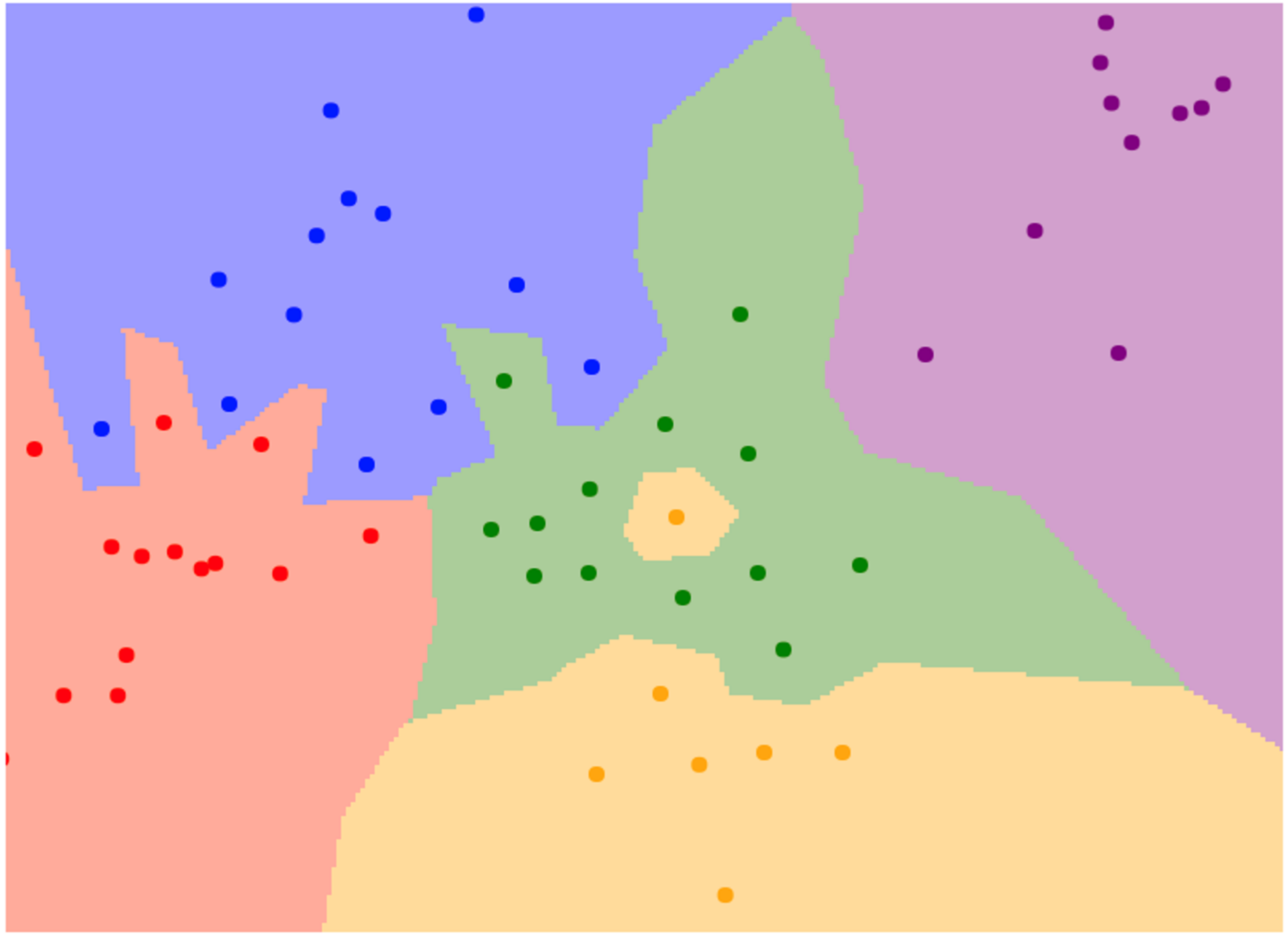
What does this look like?



What does this look like?

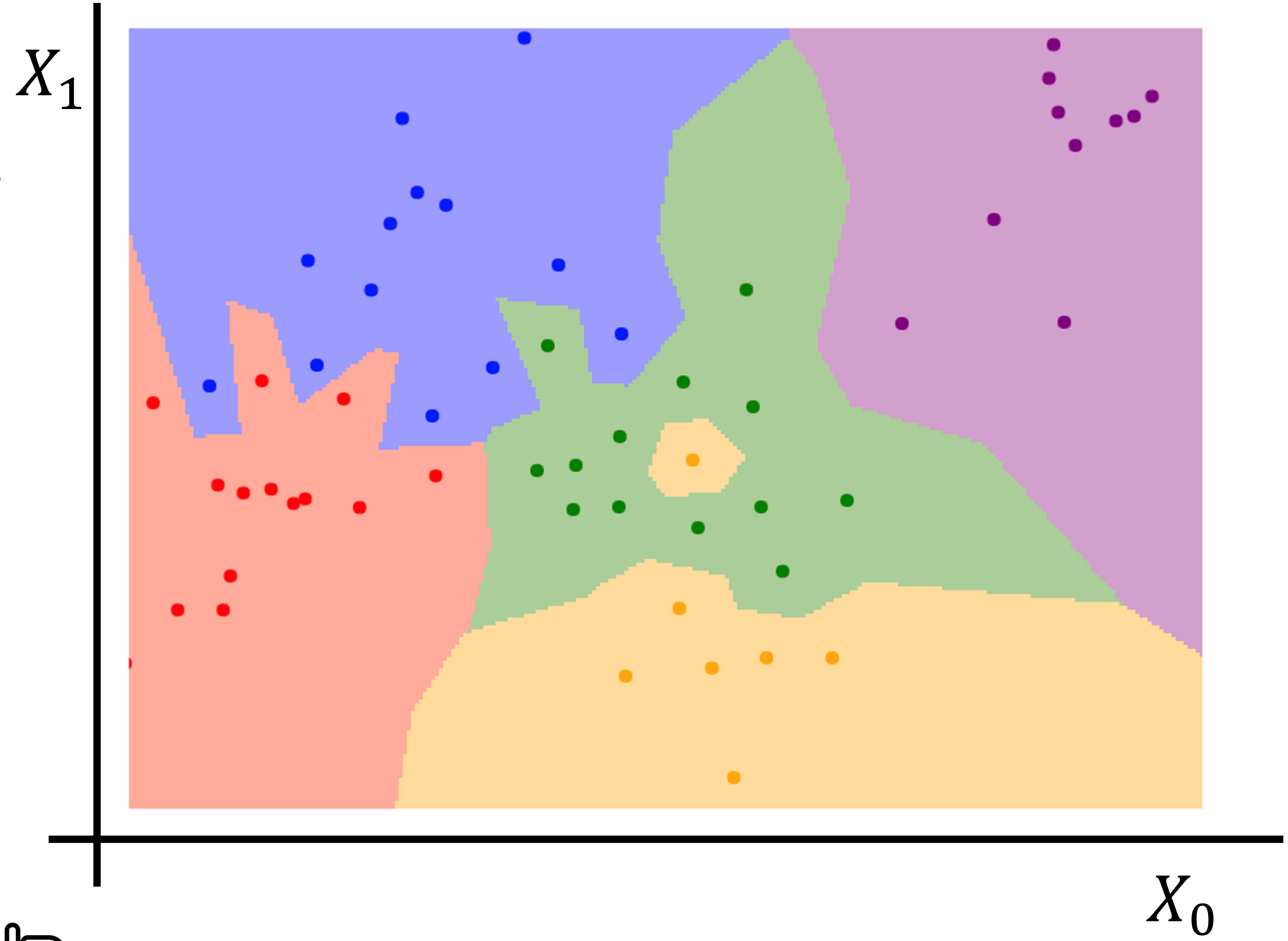
CIFAR10 dataset is category-level

K-Nearest Neighbors Decision Boundaries



K-Nearest Neighbors Decision Boundaries

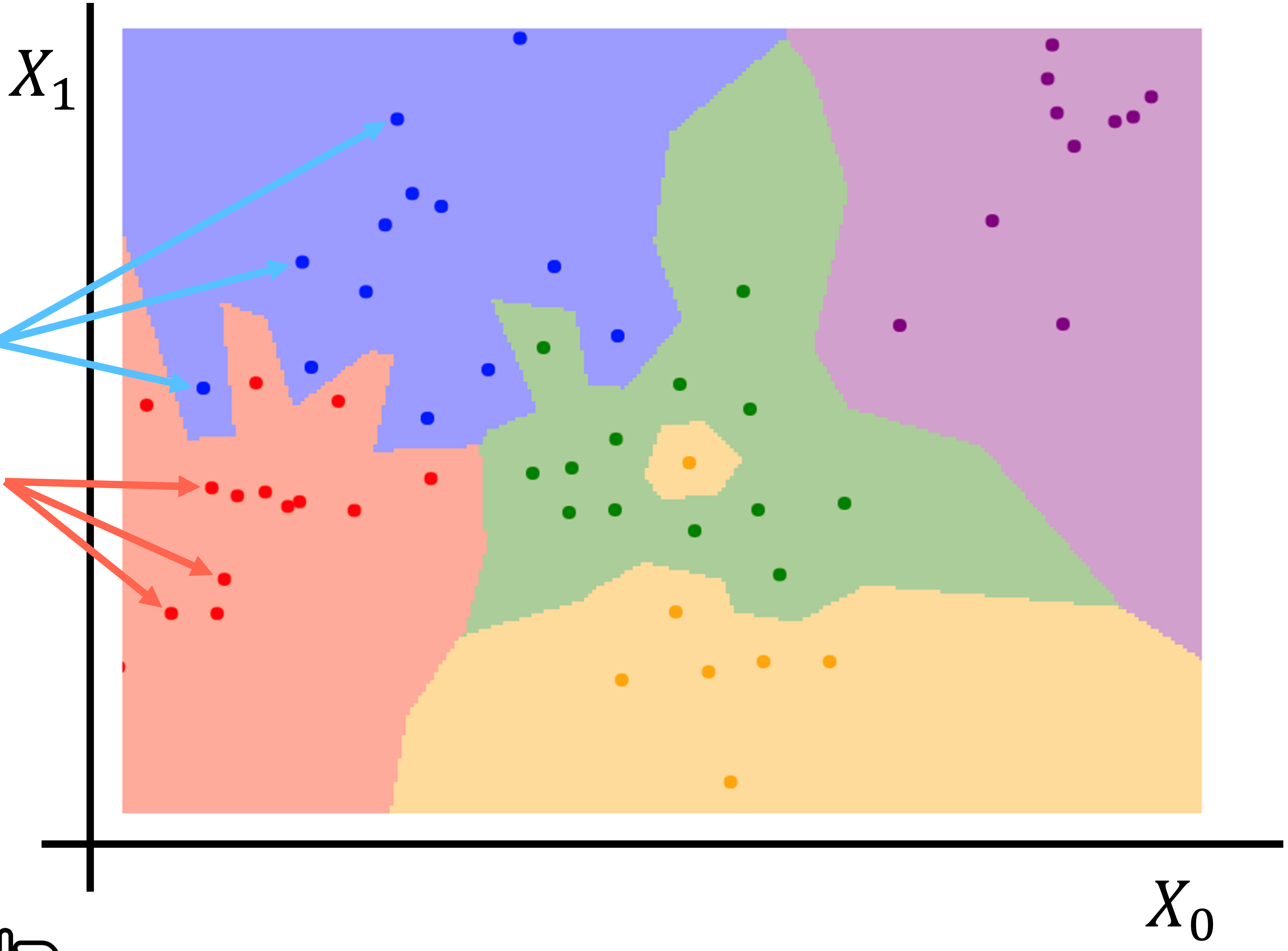
Nearest neighbors
in two dimensions



K-Nearest Neighbors Decision Boundaries

Nearest neighbors in two dimensions

Points are training examples; colors give training labels

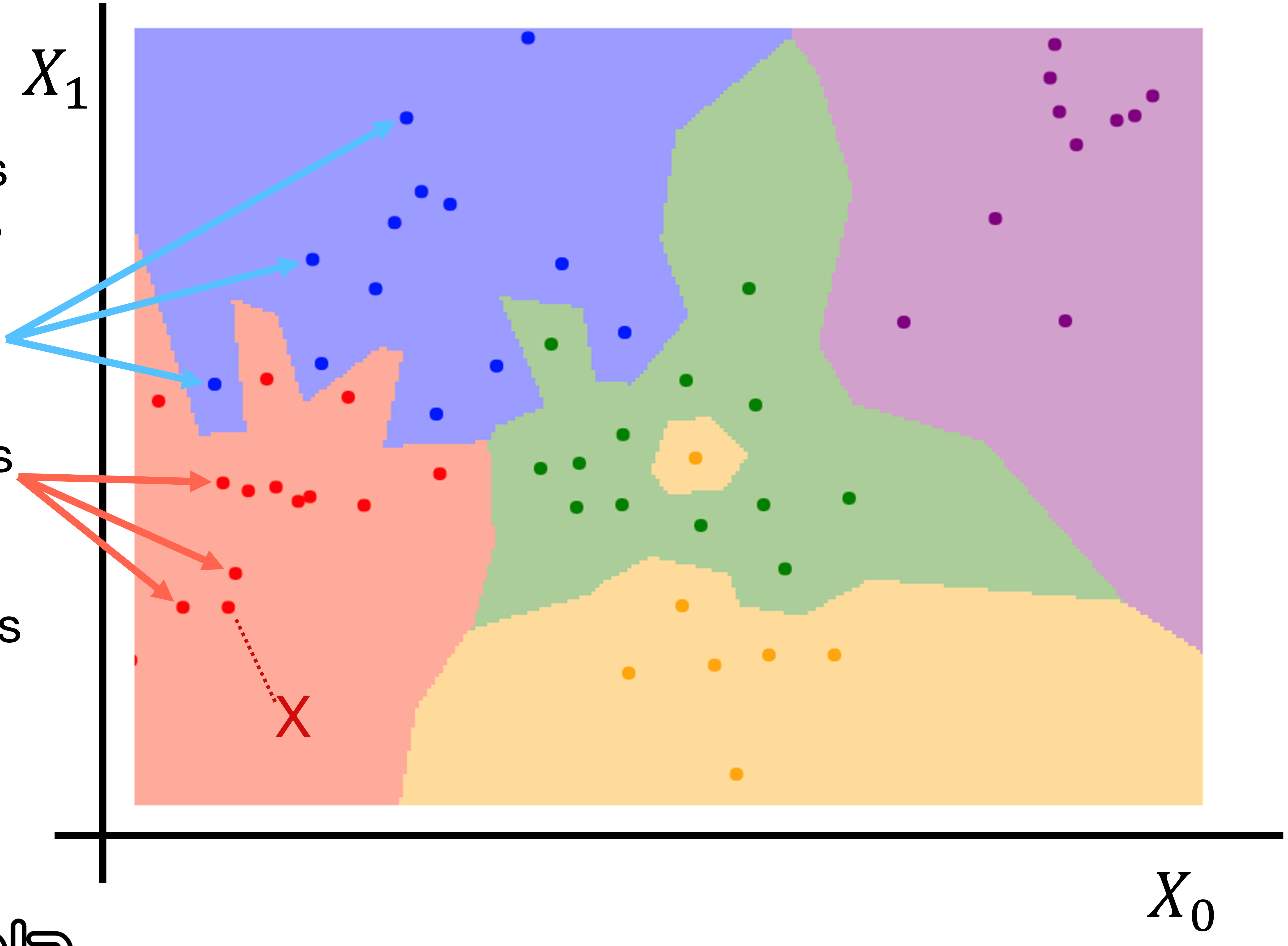


K-Nearest Neighbors Decision Boundaries

Nearest neighbors in two dimensions

Points are training examples; colors give training labels

Background colors give the category a test point would be assigned

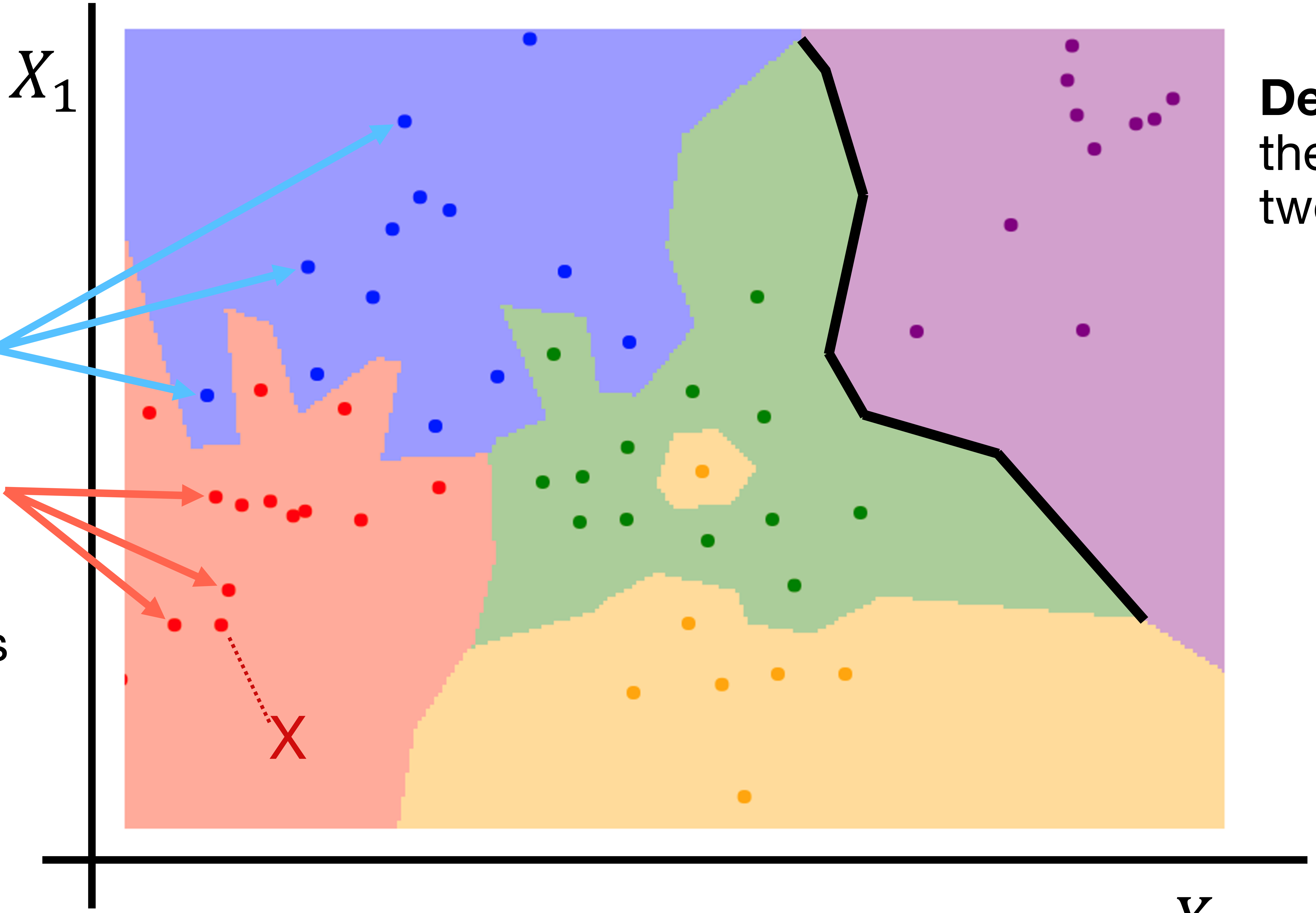


K-Nearest Neighbors Decision Boundaries

Nearest neighbors in two dimensions

Points are training examples; colors give training labels

Background colors give the category a test point would be assigned



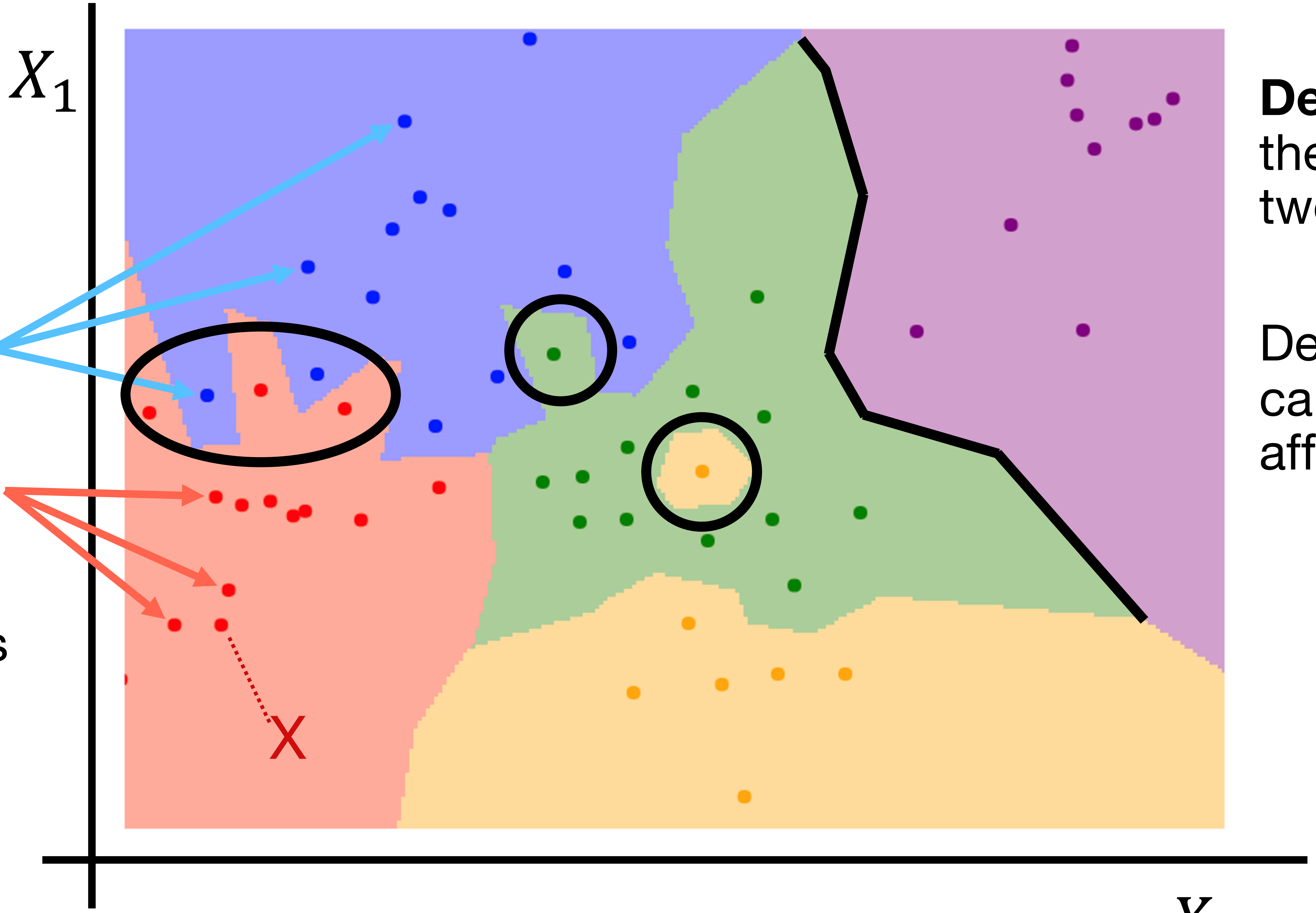
Decision boundary is the boundary between two classification regions

K-Nearest Neighbors Decision Boundaries

Nearest neighbors in two dimensions

Points are training examples; colors give training labels

Background colors give the category a test point would be assigned



Decision boundary is the boundary between two classification regions

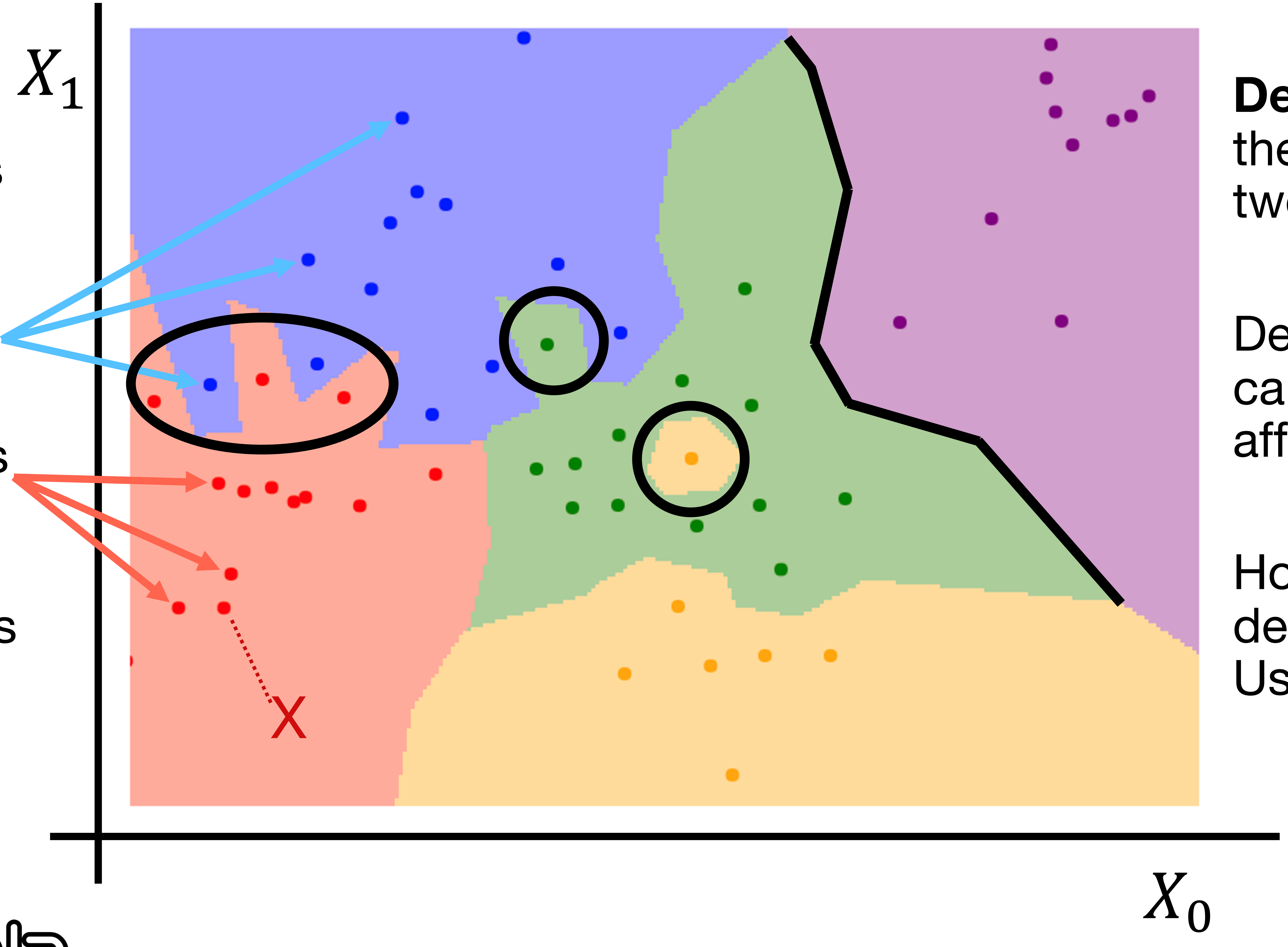
Decision boundaries can be noisy; affected by outliers

K-Nearest Neighbors Decision Boundaries

Nearest neighbors in two dimensions

Points are training examples; colors give training labels

Background colors give the category a test point would be assigned



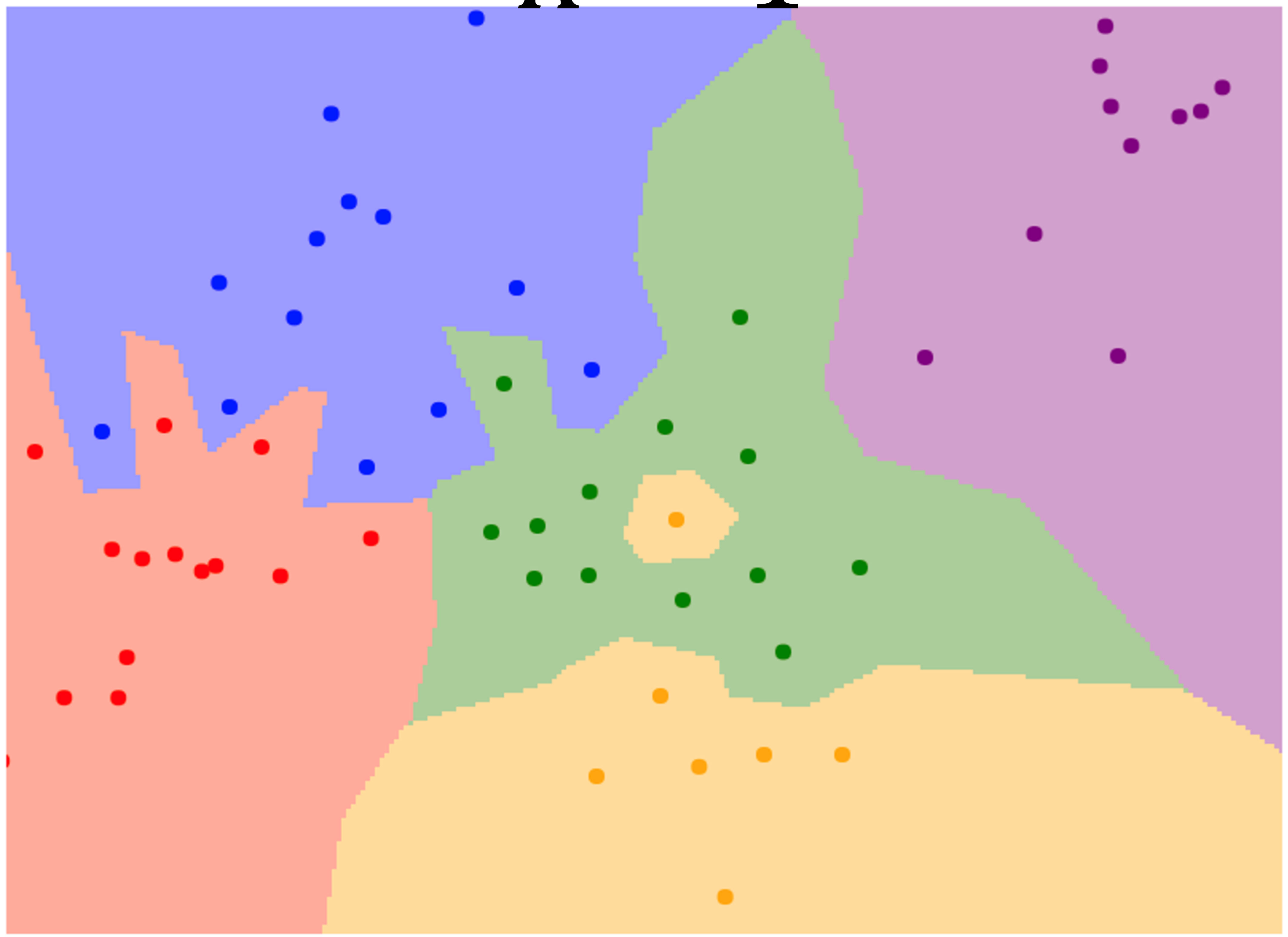
Decision boundary is the boundary between two classification regions

Decision boundaries can be noisy; affected by outliers

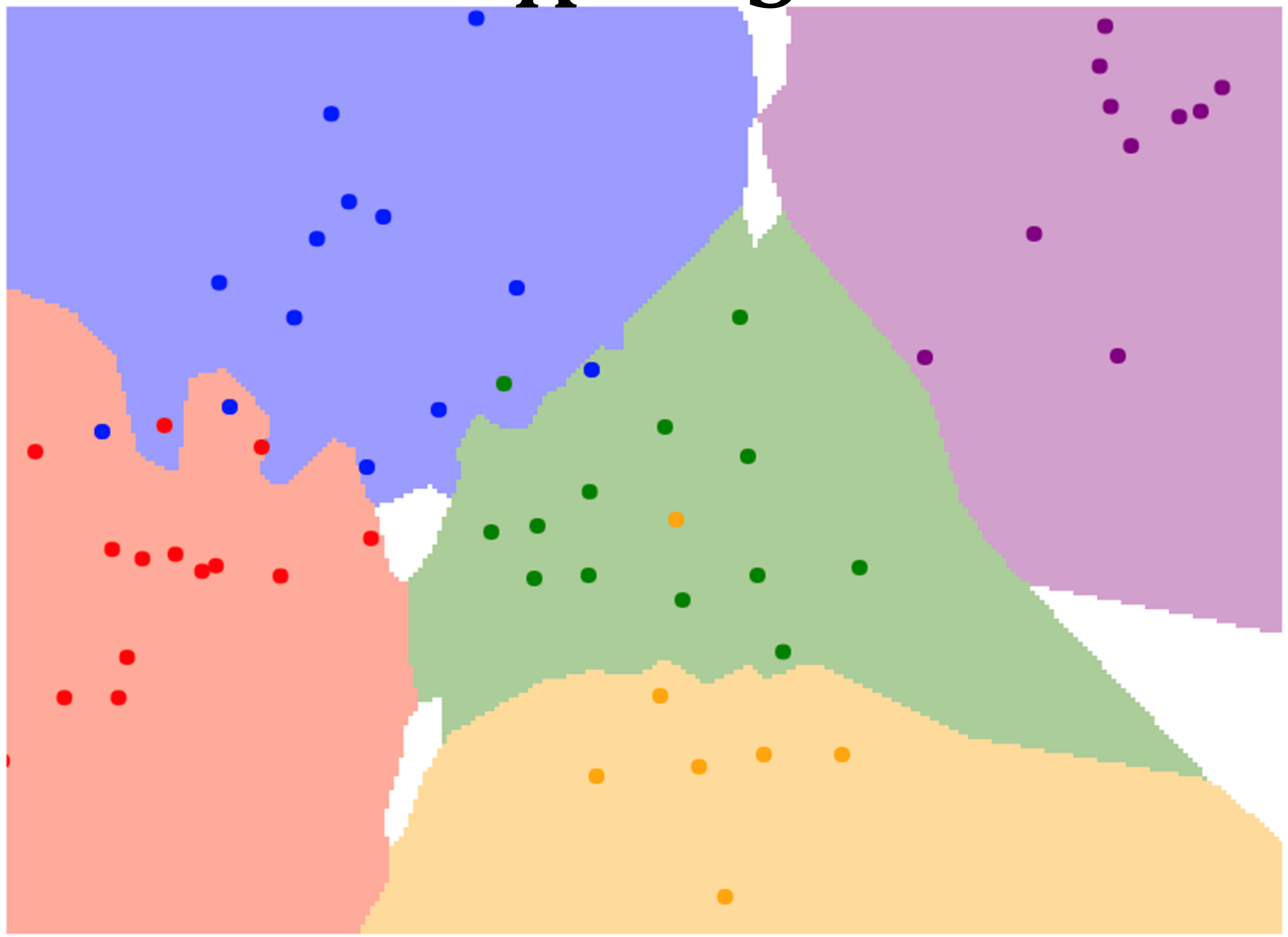
How to smooth the decision boundaries? Use more neighbors!

K-Nearest Neighbors Classification

$K = 1$



$K = 3$

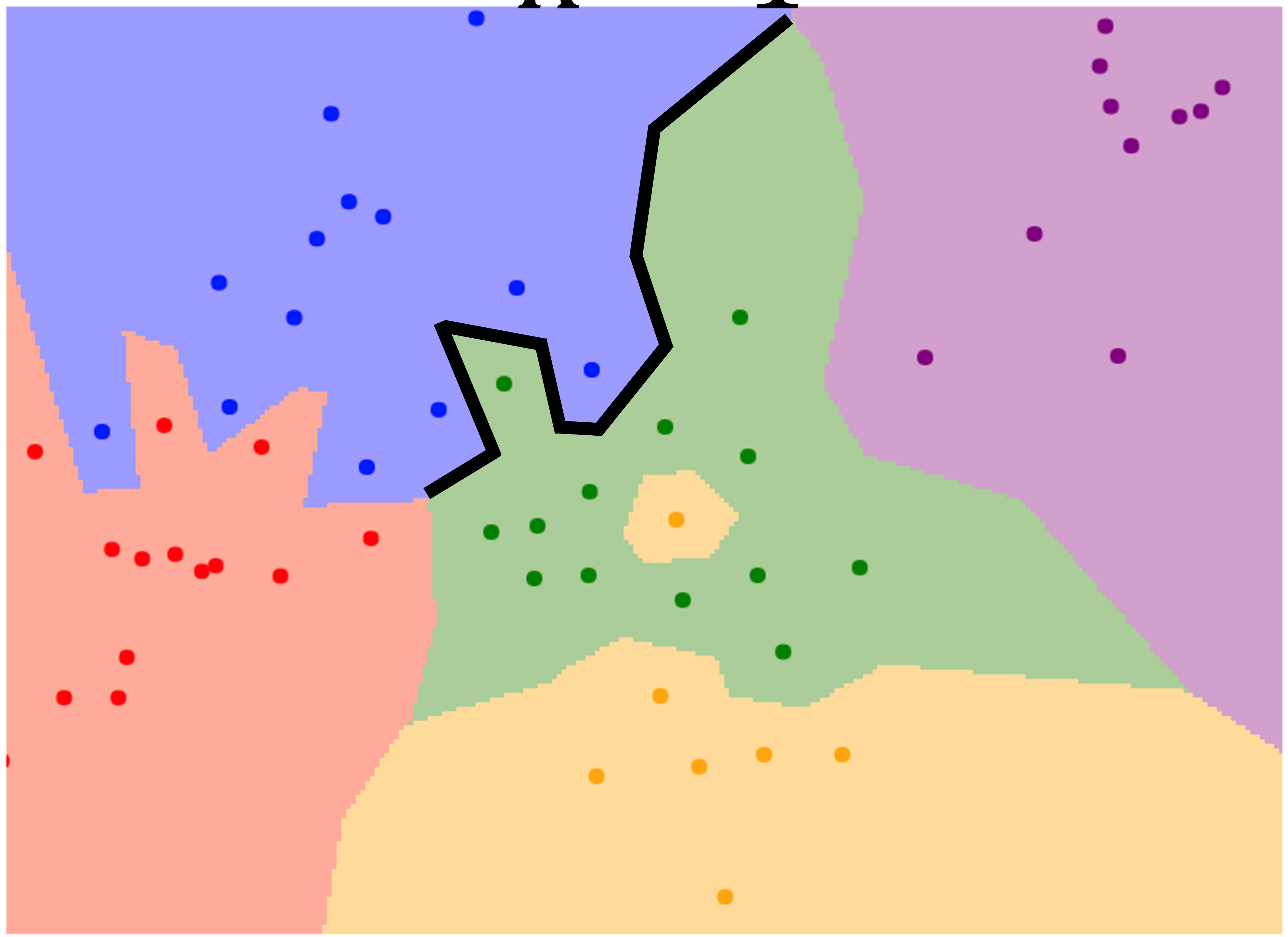


Instead of copying label from nearest neighbor,

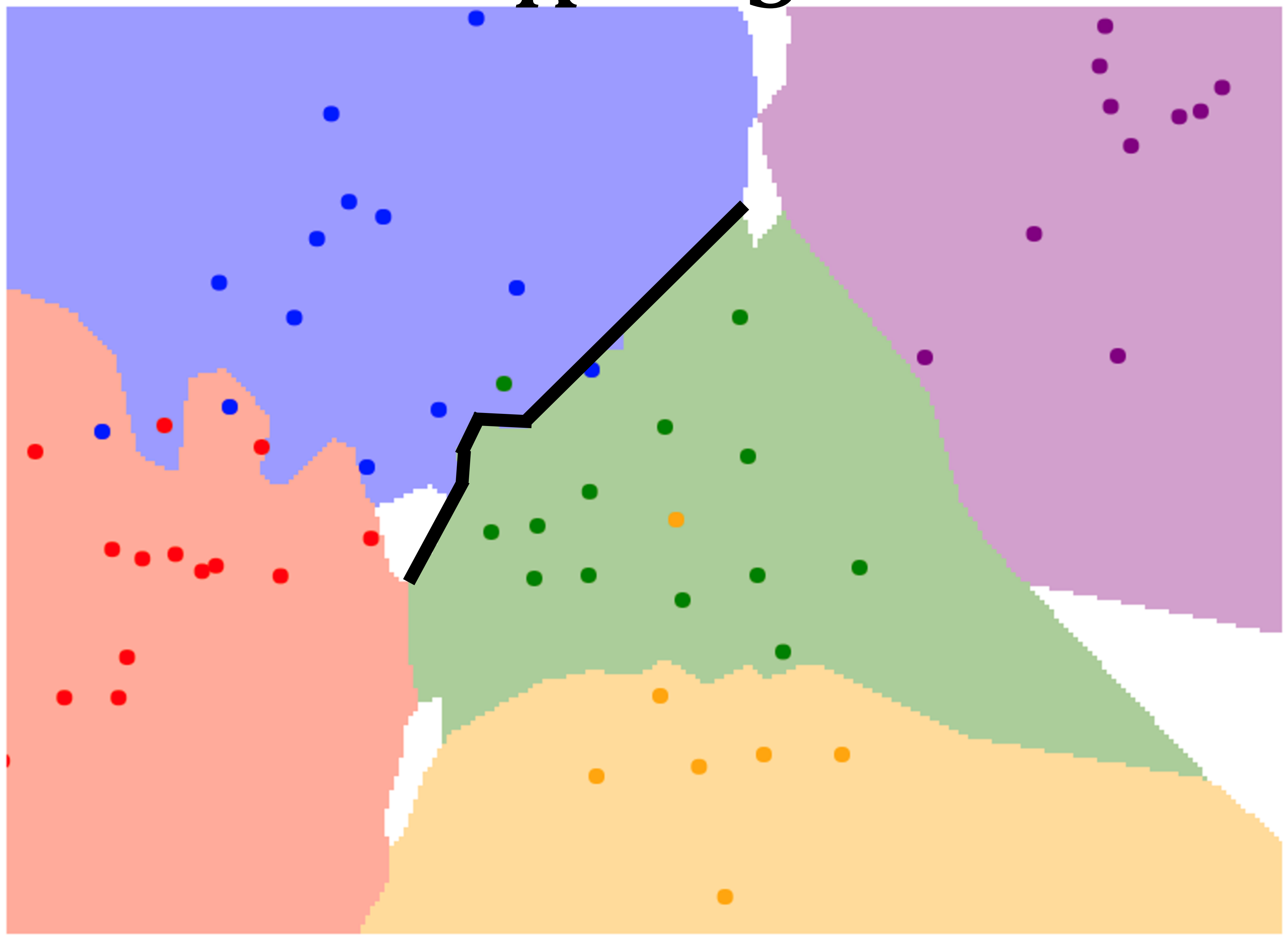
take majority vote from K closest training points

K-Nearest Neighbors Classification

$K = 1$



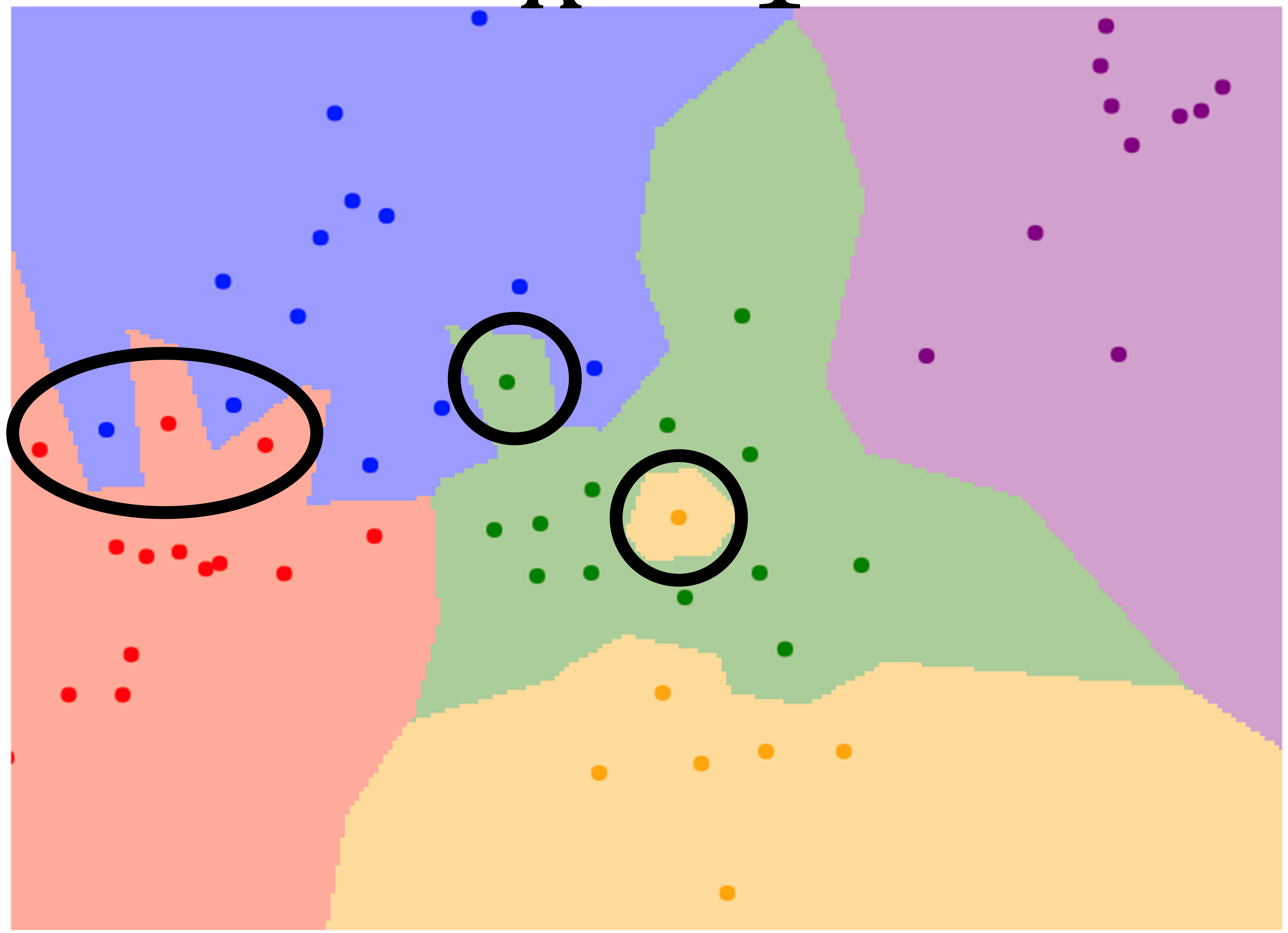
$K = 3$



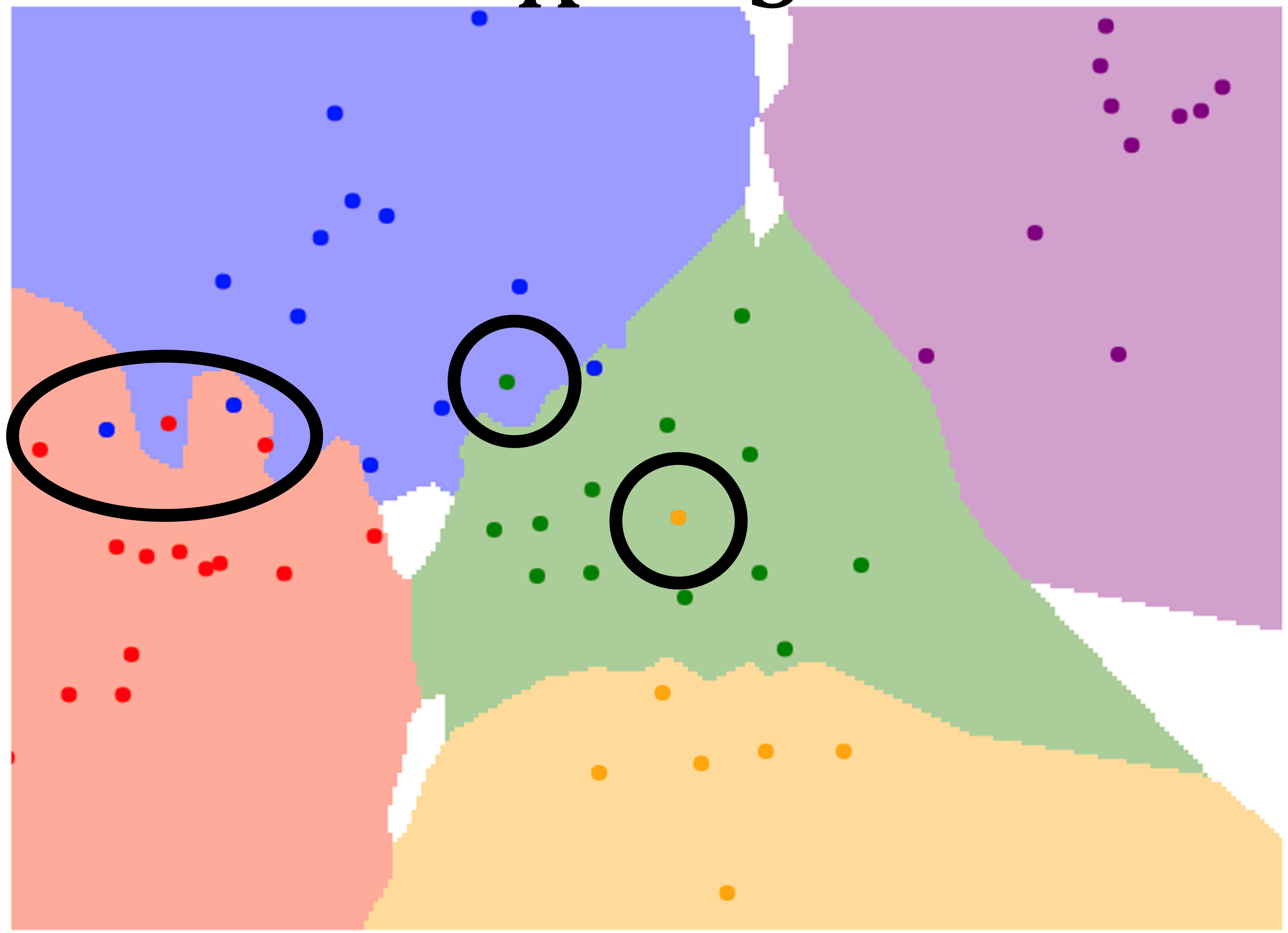
Using more neighbors helps smooth out rough decision boundaries

K-Nearest Neighbors Classification

$K = 1$



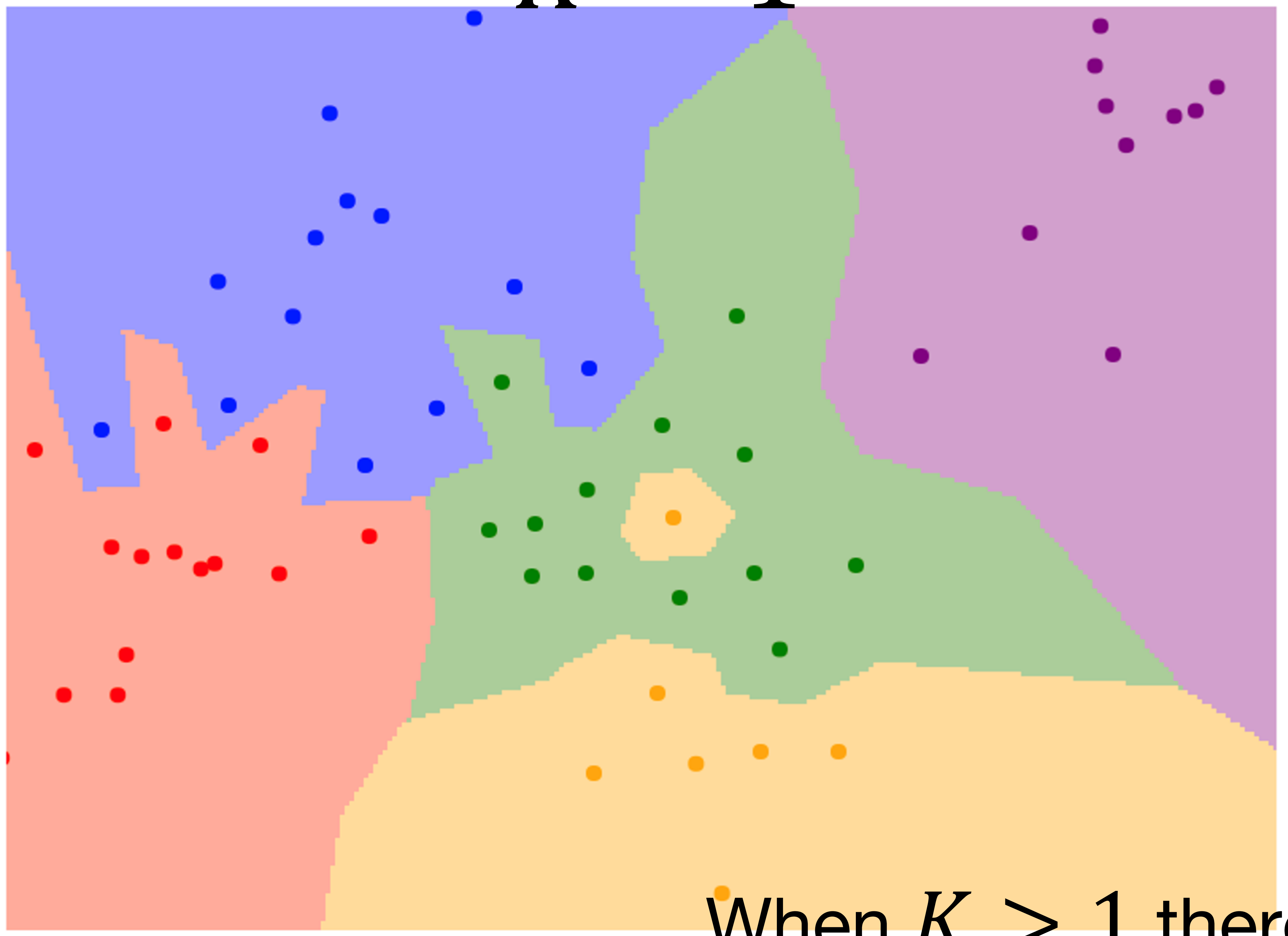
$K = 3$



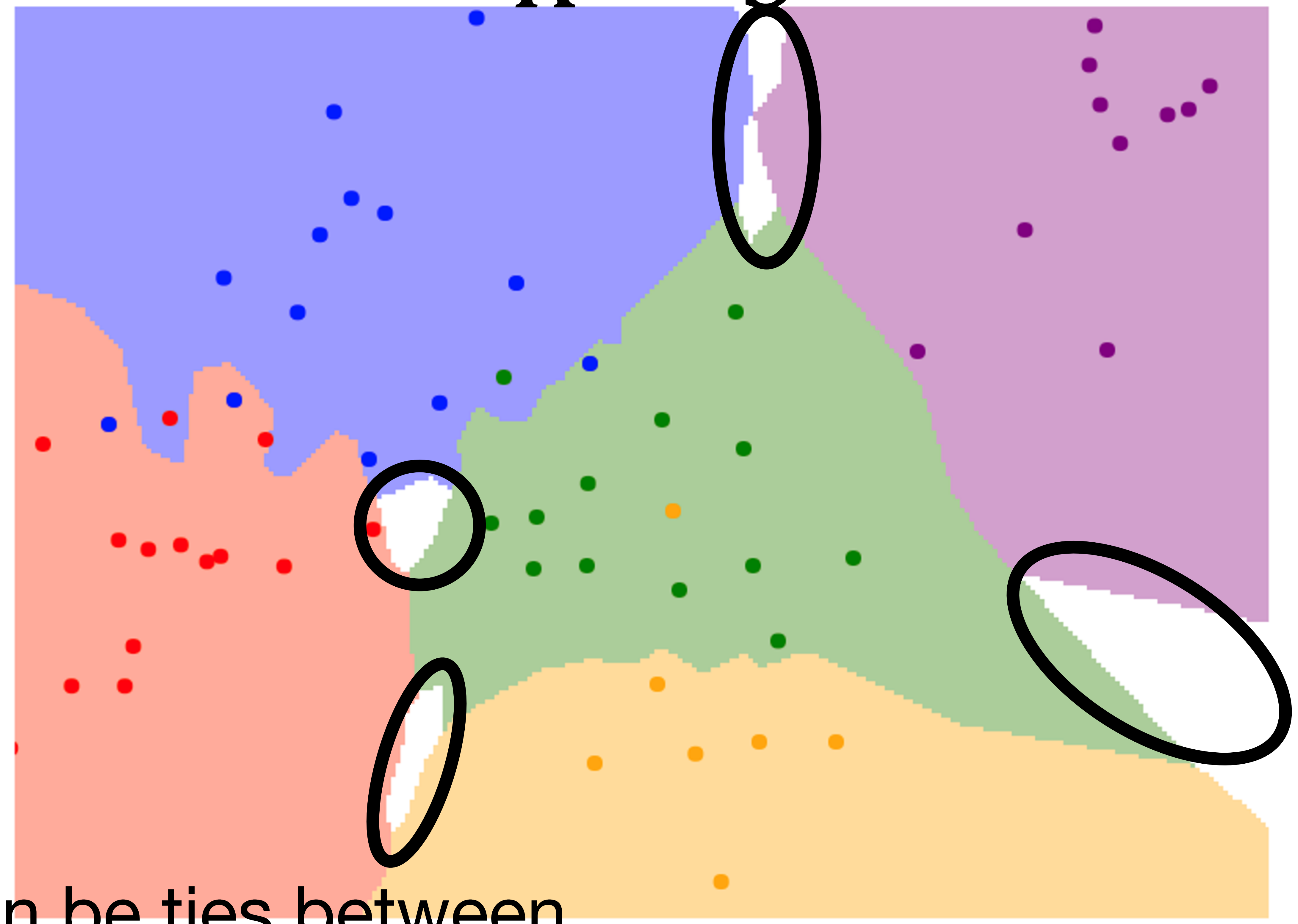
Using more neighbors helps reduce the effect of outliers

K-Nearest Neighbors Classification

$K = 1$



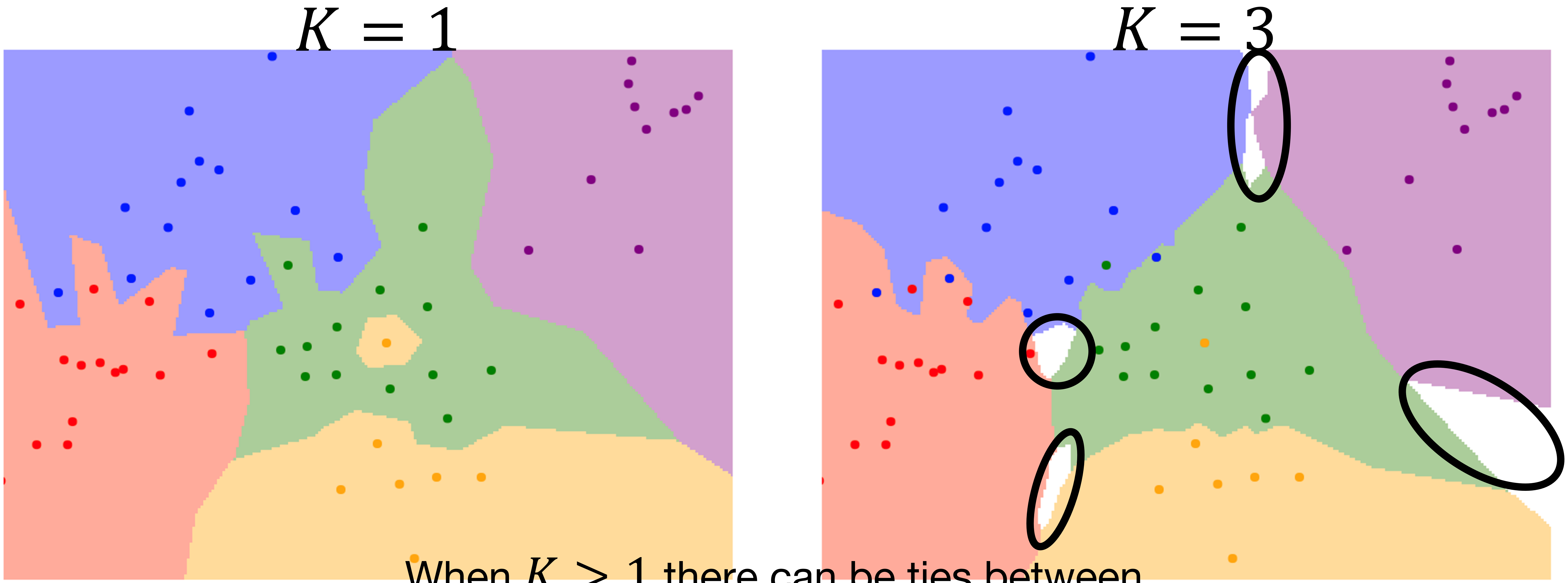
$K = 3$



When $K > 1$ there can be ties between classes.

Need to break ties somehow!

K-Nearest Neighbors Classification



When $K > 1$ there can be ties between classes.

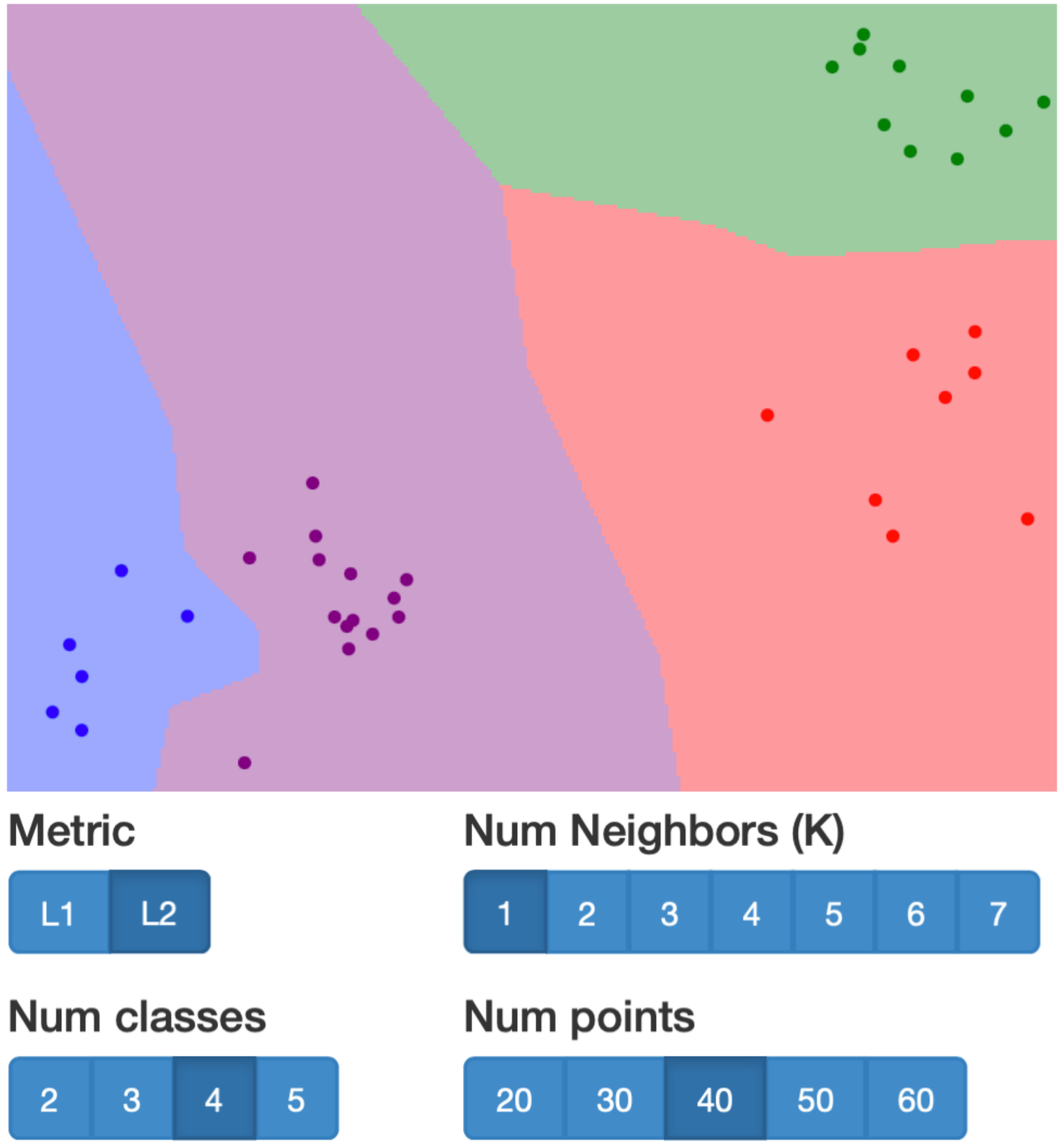
Need to break ties somehow!

K-Nearest Neighbors—Web Demo

Interactively move points around
and see decision boundaries change

Observe results with L1 vs L2 metrics

Observe results with changing number
of training points and value of K



http://vision.stanford.edu/teaching/cs231n-demos/knn/

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (Google Colab Demo)

DeepRob

Lecture 1
Image Classification
University of Michigan | Department of Robotics