CARTE ML Workshop

Class imbalance and Metric Learning

Classic Deep Learning data setup

- Classification with a single label per sample
- 2-1000 classes ; 1000+ samples per class

Classic Deep Learning data setup

- Classification with a single label per sample
- 2-1000 classes ; 1000+ samples per class

Real life problems

What if I have multiple classes per sample?

Classic Deep Learning data setup

- Classification with a single label per sample
- 2-1000 classes ; 1000+ samples per class

Real life problems

What if I have multiple classes per sample?

What if I have unbalanced or noisy labels?

Classic Deep Learning data setup

- Classification with a single label per sample
- 2-1000 classes ; 1000+ samples per class

Real life problems

What if I have multiple classes per sample?

What if I have unbalanced or noisy labels?

What if I have 1M classes and 10 samples / class?

Outline

Multi-labeling and Sampling strategies

Outline

Multi-labeling and Sampling strategies

Metric Learning and siamese networks

Outline

Multi-labeling and Sampling strategies

Metric Learning and siamese networks

Triplet Loss and advanced techniques

Multi-labeling and Sampling strategies

Dealing with Imbalance

Many real-life datasets are imbalanced, e.g. **anomaly detection**: (binary classification, most of the data is normal, very few points are anomalies)

Dealing with Imbalance

Many real-life datasets are imbalanced, e.g. **anomaly detection**: (binary classification, most of the data is normal, very few points are anomalies)

The training process may fail if most batches have only a single class in them

You may use **oversampling, importance weighting** of minority classes / samples, to help the optimization process. In Keras:

Dealing with Imbalance

Many real-life datasets are imbalanced, e.g. **anomaly detection**: (binary classification, most of the data is normal, very few points are anomalies)

The training process may fail if most batches have only a single class in them

You may use **oversampling, importance weighting** of minority classes / samples, to help the optimization process. In Keras:

```
# build your own batches and add weights
model.train_on_batch(self, x, y, class_weight=None, sample_weight=None)
```

Dealing with Imbalance

Many real-life datasets are imbalanced, e.g. **anomaly detection**: (binary classification, most of the data is normal, very few points are anomalies)

The training process may fail if most batches have only a single class in them

You may use **oversampling, importance weighting** of minority classes / samples, to help the optimization process. In Keras:

```
# build your own batches and add weights
model.train_on_batch(self, x, y, class_weight=None, sample_weight=None)
```

Evaluation must be very rigorous (the test set should represent the **true distribution** of data and labels)

Multilabel classification

Build a **binary classifier for each class** but with shared activations on hidden layers.
Easily adapted from classic classification:

```
# ...
x = Convolution2D(2048, 3)(x)
x = GlobalAveragePooling2D()(x)
x = Dense(1000, activation="softmax")(x)
```

Multilabel classification

Build a **binary classifier for each class** but with shared activations on hidden layers. Easily adapted from classic classification:

```
# ...
x = Convolution2D(2048, 3)(x)
x = GlobalAveragePooling2D()(x)
x = Dense(1000, activation="sigmoid")(x)
```

Replace `softmax` with `sigmoid` (output activation)

Replace `categorical_crossentropy` with `binary_crossentropy` (loss)

Multilabel classification

Build a **binary classifier for each class** but with shared activations on hidden layers. Easily adapted from classic classification:

```
# ...
x = Convolution2D(2048, 3)(x)
x = GlobalAveragePooling2D()(x)
x = Dense(1000, activation="sigmoid")(x)
```

Replace `softmax` with `sigmoid` (output activation)

Replace `categorical_crossentropy` with `binary_crossentropy` (loss)

y^{true} is a binary vector corresponding to classes present in the image (many-hot encoding).

Multilabel classification

Build a **binary classifier for each class** but with shared activations on hidden layers. Easily adapted from classic classification:

```
# ...
x = Convolution2D(2048, 3)(x)
x = GlobalAveragePooling2D()(x)
x = Dense(1000, activation="sigmoid")(x)
```

Replace `softmax` with `sigmoid` (output activation)

Replace `categorical_crossentropy` with `binary_crossentropy` (loss)

y^{true} is a binary vector corresponding to classes present in the image (many-hot encoding).

Problem : costly to label images exhaustively for ALL possible tags

Inference tricks

You may train with a **softmax**, and do multilabel inference with a **sigmoid**

Garrigues, P., Farfade, S., Izadinia, H., Boakye, K., & Kalantidis, Y. (2016). Tag prediction at flickr: a view from the darkroom. NIPS workshop on large scale CV 2016

Inference tricks

You may train with a **softmax**, and do multilabel inference with a **sigmoid**

Fine tune the **output bias** parameter on a small by **fully labeled** dataset

Score reflects the posterior of a tag being present in an image, e.g. $p(dog|image)$

Garrigues, P., Farfade, S., Izadinia, H., Boakye, K., & Kalantidis, Y. (2016). Tag prediction at flickr: a view from the darkroom. NIPS workshop on large scale CV 2016

Inference tricks

You may train with a **softmax**, and do multilabel inference with a **sigmoid**

Fine tune the **output bias** parameter on a small by **fully labeled** dataset

Score reflects the posterior of a tag being present in an image, e.g. $p(dog|image)$

Measure **precision/recall** per class on a fully labeled test set.

Garrigues, P., Farfade, S., Izadinia, H., Boakye, K., & Kalantidis, Y. (2016). Tag prediction at flickr: a view from the darkroom. NIPS workshop on large scale CV 2016

Metric Learning & Siamese networks

Few Examples per class

ex: Face Recognition/Verification

Few Examples per class

ex: Face Recognition/Verification

- **Recognition:** given a face, classify among K possible persons
- **Verification:** verify that two faces belongs to the same person

Few Examples per class

ex: Face Recognition/Verification

- **Recognition:** given a face, classify among K possible persons
- **Verification:** verify that two faces belongs to the same person

A verification system can be implemented as a similarity measure. If it's really good, useful for recognition.

Learning a distance function

We want to build $d_\theta(x_1, x_2)$ which indicates degree of difference between images

Learning a distance function

We want to build $d_\theta(x_1, x_2)$ which indicates degree of difference between images

We define a threshold T such that if $d_\theta(x_1, x_2) < T$, we consider x_1 and x_2 to be of the same class (e.g. identity)

Learning a distance function

We want to build $d_\theta(x_1, x_2)$ which indicates degree of difference between images

We define a threshold T such that if $d_\theta(x_1, x_2) < T$, we consider x_1 and x_2 to be of the same class (e.g. identity)

Not a real distance (no guarantee for triangle inequality)

Learning a distance function

We want to build $d_\theta(x_1, x_2)$ which indicates degree of difference between images

We define a threshold T such that if $d_\theta(x_1, x_2) < T$, we consider x_1 and x_2 to be of the same class (e.g. identity)

Not a real distance (no guarantee for triangle inequality)

Mahalanobis distance Metric Learning may be used to build distances, but are limited to linear projections, which won't be enough for fine-grained image analysis

Weinberger, Kilian Q., John Blitzer, and Lawrence K. Saul. "Distance metric learning for large margin nearest neighbor classification." Advances in neural information processing systems. 2006.

Learning a distance function

$$d_\theta(x_1, x_2) = ||f_\theta(x_1) - f_\theta(x_2)||_2$$

Learning a distance function

$$d_\theta(x_1, x_2) = ||f_\theta(x_1) - f_\theta(x_2)||_2$$

f_θ is typically a CNN which outputs a fixed dense $\mathbb{R}^d$ representation of the image.

f_θ maps image space to an **isometric representation space** where a simple metric (e.g. euclidean distance) represents **semantic distance**.

Learning a distance function

$$d_\theta(x_1, x_2) = ||f_\theta(x_1) - f_\theta(x_2)||_2$$

f_θ is typically a CNN which outputs a fixed dense $\mathbb{R}^d$ representation of the image.

f_θ maps image space to an **isometric representation space** where a simple metric (e.g. euclidean distance) represents **semantic distance**.

For other problems (sound, NLP), different networks may be used (CNN with 1D convolutions, RNNs, ...) which output a fixed dimension vector representing each input element.

Learning a distance function

$$d_\theta(x_1, x_2) = ||f_\theta(x_1) - f_\theta(x_2)||_2$$

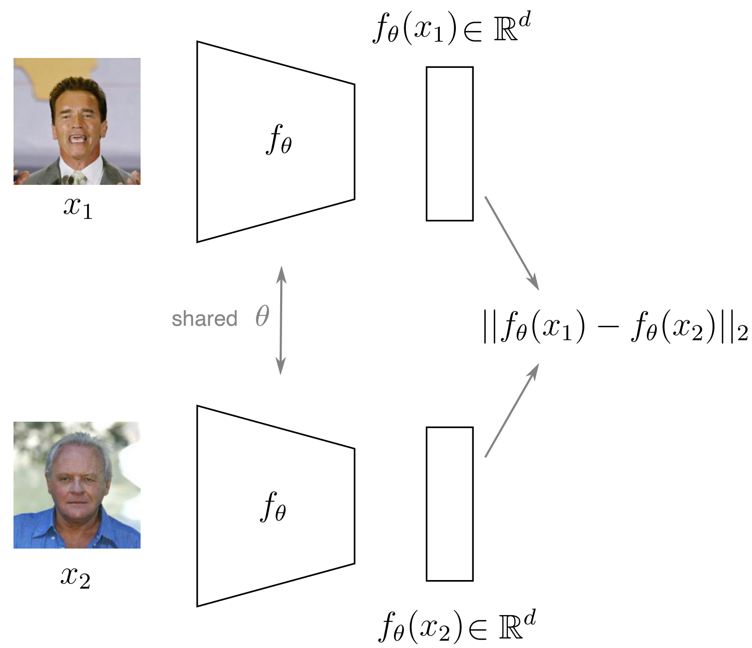
f_θ is typically a CNN which outputs a fixed dense $\mathbb{R}^d$ representation of the image.

f_θ maps image space to an **isometric representation space** where a simple metric (e.g. euclidean distance) represents **semantic distance**.

For other problems (sound, NLP), different networks may be used (CNN with 1D convolutions, RNNs, ...) which output a fixed dimension vector representing each input element.

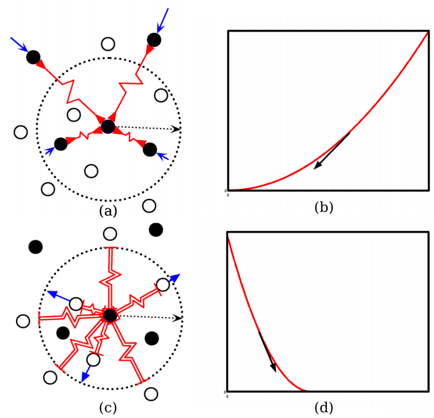
Training f_θ is also called **representation learning**

Siamese networks



Chopra, Sumit, Raia Hadsell, and Yann LeCun. "Learning a similarity metric discriminatively, with application to face verification." CVPR 2005.

Loss function



Contrastive loss: Pushes together same class pairs, and further away different class ones, up to a margin.

$$L_{\text{contrastive}}(Y, D) = \frac{1}{2}(1 - Y)D^2 + \frac{1}{2}Y \max(0, m - D)^2$$

Chopra, Sumit, Raia Hadsell, and Yann LeCun. "Learning a similarity metric discriminatively, with application to face verification." CVPR 2005.

Alternative loss functions

Can derive equivalent contrastive loss for cosine similarity:

$$cosine(f_\theta(x_1), f_\theta(x_2))$$

Alternative loss functions

Can derive equivalent contrastive loss for cosine similarity:

$$cosine(f_\theta(x_1), f_\theta(x_2))$$

Alternatively, **absolute difference + binary classification**:

$$y = sigmoid(\mathbf{w} \cdot |f_\theta(x_1) - f_\theta(x_2)| + b)$$

$y = 1$ for same class, $y = 0$ for different classes

Alternative loss functions

Can derive equivalent contrastive loss for cosine similarity:

$$cosine(f_\theta(x_1), f_\theta(x_2))$$

Alternatively, **absolute difference + binary classification**:

$$y = sigmoid(\mathbf{w} \cdot |f_\theta(x_1) - f_\theta(x_2)| + b)$$

$y = 1$ for same class, $y = 0$ for different classes

Also, **concat + binary classification**

$$y = sigmoid(\mathbf{w} \cdot f_\theta(x_1) || f_\theta(x_2) + b)$$

Alternative loss functions

Can derive equivalent contrastive loss for cosine similarity:

$$cosine(f_\theta(x_1), f_\theta(x_2))$$

Alternatively, **absolute difference + binary classification**:

$$y = sigmoid(\mathbf{w} \cdot |f_\theta(x_1) - f_\theta(x_2)| + b)$$

$y = 1$ for same class, $y = 0$ for different classes

Also, **concat + binary classification**

$$y = sigmoid(\mathbf{w} \cdot f_\theta(x_1)||f_\theta(x_2) + b)$$

Simpler:regression after cosine similarity

Training

- sample positive pairs (x_i, x_j), with (i, j) of same class
- sample negative pairs (x_i, x_j), with (i, j) of different classes

Taigman et. al., 2014. DeepFace closing the gap to human level performance

Training

- sample positive pairs (x_i, x_j), with (i, j) of same class
- sample negative pairs (x_i, x_j), with (i, j) of different classes

Forward pass using both inputs through the two networks (**same parameters, different activations**)

Taigman et. al., 2014. DeepFace closing the gap to human level performance

Training

- sample positive pairs (x_i, x_j), with (i, j) of same class
- sample negative pairs (x_i, x_j), with (i, j) of different classes

Forward pass using both inputs through the two networks (**same parameters, different activations**)

Backpropagate through the two networks (the weights are updated with the **sum of the two gradients**)

Taigman et. al., 2014. DeepFace closing the gap to human level performance

Training

- sample positive pairs (x_i, x_j), with (i, j) of same class
- sample negative pairs (x_i, x_j), with (i, j) of different classes

Forward pass using both inputs through the two networks (**same parameters, different activations**)

Backpropagate through the two networks (the weights are updated with the **sum of the two gradients**)

Important to **craft batches carefully** (balance positive and negative, group positives together). Many negatives are *easy* (closer than margin) & don't contribute to the loss.

Taigman et. al., 2014. DeepFace closing the gap to human level performance

Dataset Face verification

Labeled Faces in the Wild (LFW)

13,000 pictures, 1680 identities with at least 2 pictures

Dataset Face verification

Labeled Faces in the Wild (LFW)

13,000 pictures, 1680 identities with at least 2 pictures

- results http://vis-www.cs.umass.edu/lfw/results.html
- DeepFace: 0.9735 binary classification accuracy

Taigman et. al., 2014. DeepFace closing the gap to human level performance

Dataset Face verification

Labeled Faces in the Wild (LFW)

13,000 pictures, 1680 identities with at least 2 pictures

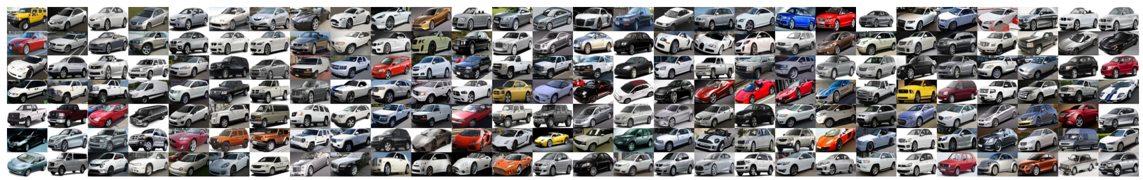
- results http://vis-www.cs.umass.edu/lfw/results.html
- DeepFace: 0.9735 binary classification accuracy

YouTube Faces Database

3,425 videos of 1,595 different people averaging 181 frames per video

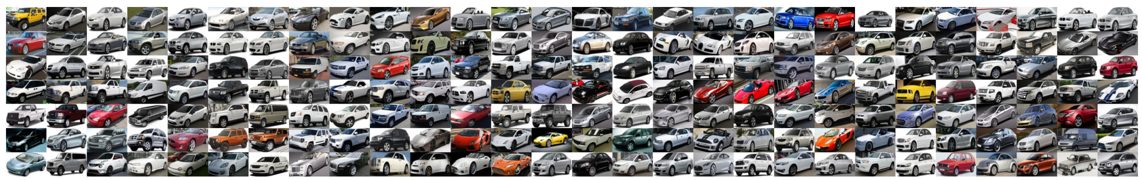
Taigman et. al., 2014. DeepFace closing the gap to human level performance

Cars196, CUB200, Online Products



16,185 images of 196 classes of cars

Cars196, CUB200, Online Products



16,185 images of 196 classes of cars

120,053 images, 22,634 Online Products (classes) from eBay.com. 5.3 images per class

Triplet Loss

Triplet Loss

A triplet: (x^a, x^+, x^-)

- an anchor image
- a positive image (same person as anchor)
- a negative image (different person as anchor)

We compute f_θ for each of these 3 images

Triplet Loss

A triplet: (x^a, x^+, x^-)

- an anchor image
- a positive image (same person as anchor)
- a negative image (different person as anchor)

We compute f_θ for each of these 3 images

We want that:

$$||f(x^a) - f(x^-)||_2 \geq ||f(x^a) - f(x^+)||_2$$

Triplet Loss

A triplet: (x^a, x^+, x^-)

- an anchor image
- a positive image (same person as anchor)
- a negative image (different person as anchor)

We compute f_θ for each of these 3 images

We want that:

$$||f(x^a) - f(x^-)||_2 \geq ||f(x^a) - f(x^+)||_2$$

$$||f(x^a) - f(x^-)||_2 - ||f(x^a) - f(x^+)||_2 \geq \alpha$$

α **margin** typically small positive number $(0.2, 0.5, \cdot)$

Triplet Loss

A triplet: (x^a, x^+, x^-)

- an anchor image
- a positive image (same person as anchor)
- a negative image (different person as anchor)

We compute f_θ for each of these 3 images

We want that:

$$||f(x^a) - f(x^-)||_2 \geq ||f(x^a) - f(x^+)||_2$$

$$||f(x^a) - f(x^-)||_2 - ||f(x^a) - f(x^+)||_2 \geq \alpha$$

α **margin** typically small positive number $(0.2, 0.5, \cdot)$

minimize $||f(x^a) - f(x^+)||_2 - ||f(x^a) - f(x^-)||_2 + \alpha$

Triplet Loss

$l(x^a, x^+, x^-) =$

$max(||f(x^a) - f(x^+)||_2 - ||f(x^a) - f(x^-)||_2 + \alpha, 0)$

Schroff et al.,2015, FaceNet: A unified embedding for face recognition and clustering

Triplet Loss

$l(x^a, x^+, x^-) =$

$max(||f(x^a) - f(x^+)||_2 - ||f(x^a) - f(x^-)||_2 + \alpha, 0)$

Training

- Sample a minibatch of triplets $\{(x^a, x^+, x^-)_i\}$
- Forward pass on all 3 networks using f_θ
- Compute $\sum_i l((x^a, x^+, x^-)_i)$

Schroff et al.,2015, FaceNet: A unified embedding for face recognition and clustering

Triplet Loss

$l(x^a, x^+, x^-) =$

$max(||f(x^a) - f(x^+)||_2 - ||f(x^a) - f(x^-)||_2 + \alpha, 0)$

Training

- Sample a minibatch of triplets $\{(x^a, x^+, x^-)_i\}$
- Forward pass on all 3 networks using f_θ
- Compute $\sum_i l((x^a, x^+, x^-)_i)$
- Compute the gradients through the 3 networks
- update f_θ using the sum of 3 gradients

Schroff et al.,2015, FaceNet: A unified embedding for face recognition and clustering

Hard negative sampling

After a few epochs, If (x^a, x^+, x^-) are chosen randomly, it will be easy to satisfy the inequality.

Hard negative sampling

After a few epochs, If (x^a, x^+, x^-) are chosen randomly, it will be easy to satisfy the inequality.

Gradient in one batch quickly become almost 0 except for **hard cases**. Random sampling is inefficient to find these hard cases

Hard negative sampling

After a few epochs, If (x^a, x^+, x^-) are chosen randomly, it will be easy to satisfy the inequality.

Gradient in one batch quickly become almost 0 except for **hard cases**. Random sampling is inefficient to find these hard cases

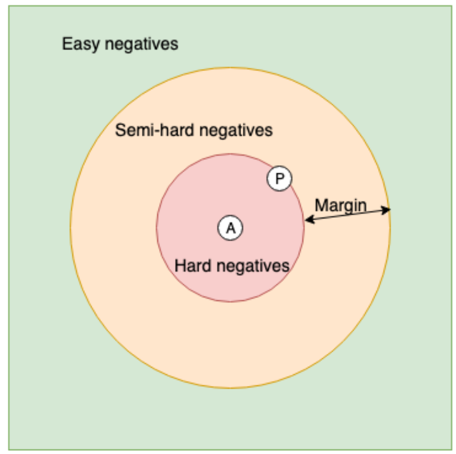
- **Hard triplet sampling:** sample x^- such that:

$$||f(x^a) - f(x^+)||_2 > ||f(x^a) - f(x^-)||_2 + \alpha$$

- **Semi Hard triplet sampling:** sample x^- such that:

$$||f(x^a) - f(x^+)||_2 > ||f(x^a) - f(x^-)||_2$$

Hard negative sampling



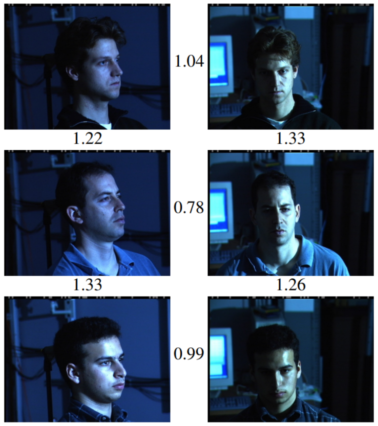
- **Hard triplet sampling:** sample x^- such that:

$$||f(x^a) - f(x^+)||_2 > ||f(x^a) - f(x^-)||_2 + \alpha$$

- **Semi Hard triplet sampling:** sample x^- such that:

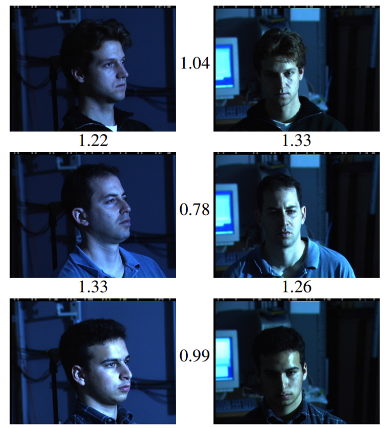
$$||f(x^a) - f(x^+)||_2 > ||f(x^a) - f(x^-)||_2$$

Triplets results



Schroff, Florian, et al. Facenet: A unified embedding for face recognition and clustering, CVPR 2015. = hard negative mining and semi hard

Triplets results



- A threshold is computed on test set (1.2)
- Best model achieves 99.6% verification accuracy on LFW
- Face alignment is critical!

Schroff, Florian, et al. Facenet: A unified embedding for face recognition and clustering, CVPR 2015. = hard negative mining and semi hard

Model in production

Deploying Verification models e.g. Iphone FaceID:

Build as light as possible model, both in terms of memory footprint and computing cost

Model in production

Deploying Verification models e.g. Iphone FaceID:

Build as light as possible model, both in terms of memory footprint and computing cost

Ask the user 10 photos and precompute representations of these.

Model in production

Deploying Verification models e.g. Iphone FaceID:

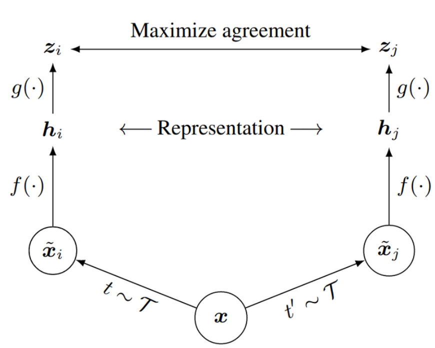
Build as light as possible model, both in terms of memory footprint and computing cost

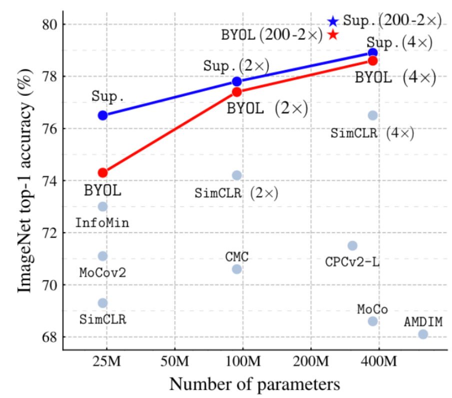
Ask the user 10 photos and precompute representations of these.

At test time, compute representation of the new photo, then compute similarities with the 10 representations

If a similarity is within a predefined threshold then Unlock!

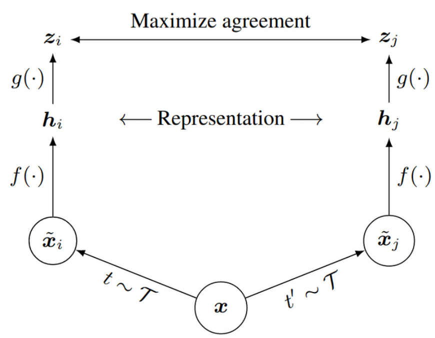
Self-supervised learning

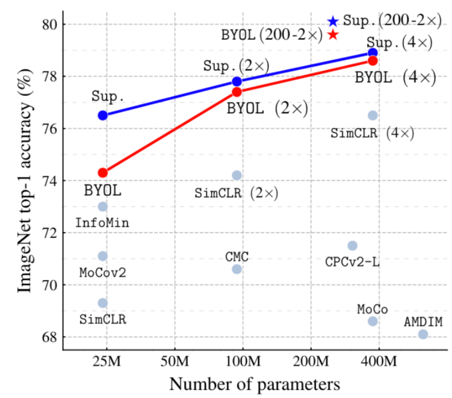




SimCLR uses a contrastive loss on pairs of heavily augmented images vs independent images in the batch.

Self-supervised learning





[SimCLR]() uses a contrastive loss on pairs of heavily augmented images vs independent images in the batch.

[BYOL]() and [data2vec]() can even do away with the negative terms.

Take Aways on classification

- For most cases, use a **classifier with softmax**
- If you have many classes, and/or strong class imbalance use constrastive-based **metric learning**
- Both can give useful embeddings for clustering / low shot learning

Take Aways on classification

- For most cases, use a **classifier with softmax**
- If you have many classes, and/or strong class imbalance use constrastive-based **metric learning**
- Both can give useful embeddings for clustering / low shot learning

- Use **ImageNet pre-trained features** in most cases
- If you have many many unlabeled images, self-supervised learning can be an alternative
- Very active area of research!

Thank you!