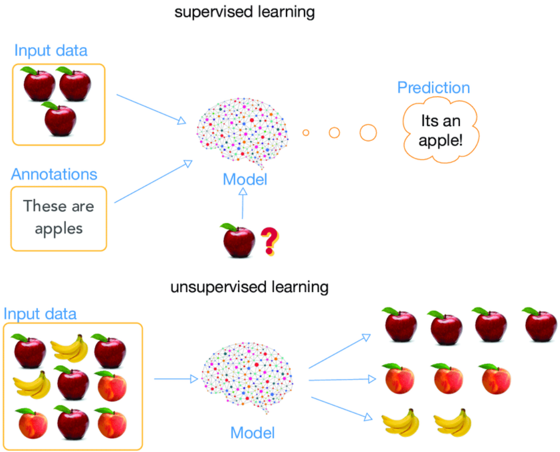
# Self supervised learning with Contrastive Loss

Fanta, Amel, Allassan, Elabbas

African Institute for Mathematical Sciences, AIMS-Senegal
Supervised by: Dr Moustapha CISSE
African Master's of Machine Intelligence
AIMS | African Institute for Mathematical Sciences SENEGAL

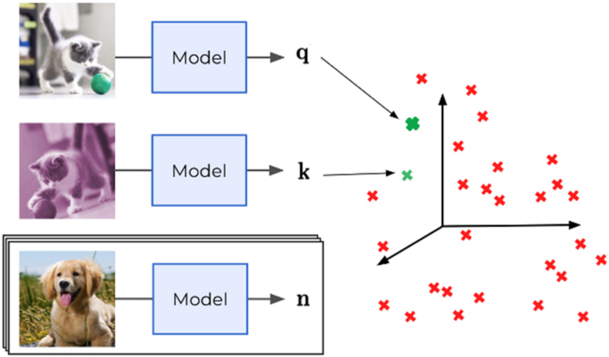
# Outline

# Details




[source]

# Details



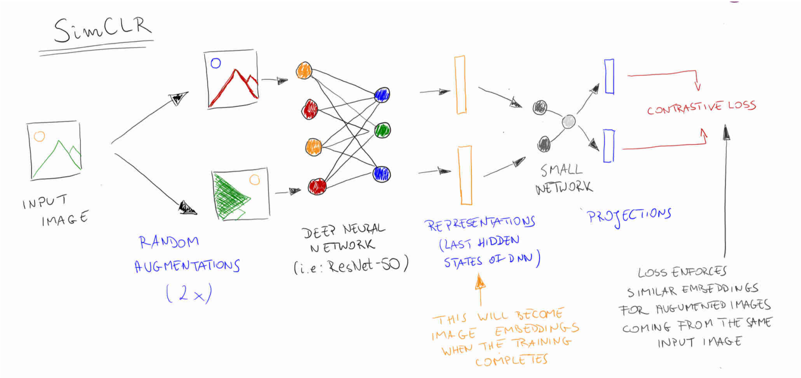

[source]

# Self-supervised vs. supervised learning

Self-supervised learning is different to supervised learning in that :

- It's a form of unsupervised learning where the data provides the supervision
- In self-supervised training we use some kind of measurable structure to build a loss function

# SimCLR Framework




[source]

# General terms

## Definitions

- Similarity function : A measure of how close or far two embeddings are from each other, in this case we will be using cosine similarity, denoted as $sim(x_i, x_j)$

$$Cosine\_similarity = \frac{A.B}{\|A\|\|B\|} \tag{1}$$

- Embedding vector : A representation of the input data in a vectorized form, denoted $x_i$
- Positive pairs : Embeddings from the same input source
- Negative pairs : Embeddings from different input sources
- Temperature : Term controlling the strength of penalties on the hard negatives, denoted $\tau$

# General terms

## Definitions

For two augmented views $x_i$ and $x_j$ (positive pairs), of the same input image $x$, let's consider their embeddings after projection to be $z_i$ and $z_j$. The contrastive loss function associated to this positive pair is given by :

$$l_{(i,j)} = -log\left(\frac{exp\left(\frac{sim(z_i,z_j)}{\tau}\right)}{\sum_{k=1}^{2N}\mathbb{1}_{\{k\neq i\}}exp\left(\frac{sim(z_i,z_k)}{\tau}\right)}\right). \quad (2)$$

# General terms

## Definitions

The final loss is an arithmetic mean of the losses for all positive pairs in the batch :

$$L = \frac{1}{2N} \sum_{k=1}^{2N} \left( l_{(2k-1,2k)} + l_{(2k,2k-1)} \right) . \tag{3}$$

# Algorithm

**Algorithm 1** SimCLR's main learning algorithm.

**input:** batch size $N$, constant $\tau$, structure of $f$, $g$, $\mathcal{T}$.
**for** sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  **for all** $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  **end for**
  **for all** $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  **end for**
  **define** $\ell(i,j)$ **as** $\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks $f$ and $g$ to minimize $\mathcal{L}$
**end for**
**return** encoder network $f(\cdot)$, and throw away $g(\cdot)$

[A Simple Framework for Contrastive Learning of Visual Representations 2020]

1. Machine learning methods
   - Supervised and unsupervised learning
   - Self supervised learning
2. SimCLR
   - Understanding SimCLR
   - Contrastive loss
3. **Implementation details**
   - Normal image classification
   - Self-supervised task
4. Results and discussions

# Supervised task

## Data Augmentation

We used the *CIFAR*10 data set and performed the following data augmentation techniques :

- Random cropping and resizing ;
- Horizontal Flipping ;
- Normalization.

# Supervised task

## Experimental set-up

We used $50,000$ images for training and $10,000$ images for testing following the configuration below

- Model : Resnet50
- Optimizer : Adam
- Batch size : 64

The loss function used was the cross-entropy globally defined as for N-classes :

$$CE_{loss} = -log\left(\frac{e^{x_i}}{\sum_{c=1}^{N} e^{x_c}}\right). \tag{4}$$

# Embeddings Generation

## Experimental Settings

- We experimented with both LARS and Adam optimizer with a batch size of 64, more details are in section 4.

## Data Augmentation

simCLR uses the following augmentations to produce image views :

- Random cropping and resizing
- Random horizontal flip
- Gaussian blur*
- Color jitter

# Embeddings Generation

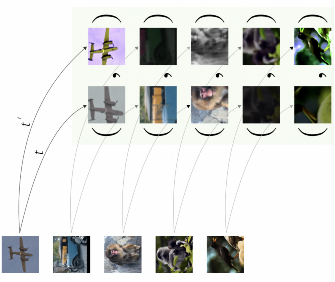



Figure 1 – Positive pairs generation

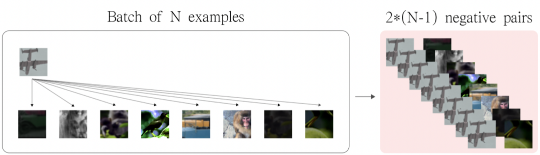



Figure 2 – Negative pairs generation

[source]

# Results and discussions

| Model | Optimizer | lr | num_epochs | accuracy |
|---|---|---|---|---|
| SSL | LARS | 0.6 | 10 | - |
| Fine-Tuning | Adam | 0.001 | 10 | 31% |
| SSL | LARS | 0.6 | 10 | - |
| Fine-Tuning | Adam | 0.001 | 20 | 33% |
| SSL | LARS | 0.6 | 10 | - |
| Fine-Tuning | Adam | 0.1 | 10 | 17% |
| SSL | Adam | 0.075 | 10 | - |
| Fine-Tuning | Adam | 0.01 | 10 | 36% |
| SSL | Adam | 0.075 | 10 | - |
| Fine-Tuning | Adam | 0.01 | 20 | 35% |
| SL | Adam | 0.0001 | 10 | 20% |

SSL $\equiv$ *Self_Supervised_Learning*
*SL* $\equiv$ *Supervised_Learning*

Thank You for Tuning in !

Any Questions ?