- #paper/to-read ~ 2016 CE ~ Metric Learning, Face Recognition, Loss Function
  - **A Discriminative Feature Learning Approach for Deep Face Recognition**
  - https://ydwen.github.io/papers/WenECCV16.pdf
  - Mentioned topics:
    - Siamese Networks

- # Summary

  - For each sample in a mini-batch, an Embedding is generated. Then the centroids are calculated for the embeddings corresponding to each class. Finally, the distances from embeddings to their class centroids are added to the loss value.
  - The final loss is a balanced sum of a classification loss (usually Softmax) and the center loss:

$$\mathcal{L} = \mathcal{L}_{Softmax} + \lambda \mathcal{L}_{Center}$$

$$\mathcal{L}_{Center} = \frac{1}{2} \sum_{i=1}^{m} ||x_i - c_{y_i}||_2^2$$

    - The $c_{y_i} \in \mathbb{R}^d$ denotes the y_ith class center of deep features (i.e. Embeddings).
      - The update of c_j is computed as:

$$\Delta c_j = \frac{\sum_{i=1}^{m} \mathbb{1}(y_i = j) \cdot (c_j - x_i)}{1 + \sum_{i=1}^{m} \mathbb{1}(y_i = j)}$$

$$c_j^{t+1} = c_j^t - \alpha \cdot \Delta c_j^t$$

  - The Training Process is much more stable for the center loss than it is for Triplet Loss or Contrastive Loss, mostly due to the fact that the size of a dataset doesn't inflate.

- # Ideas

- Train a model on 7-8 classes from MNIST.
- In a model that outputs Embeddings, apply the center loss to an intermediate embedding that is more high-dimensional than the output one.
    - Or, probably, apply it to both of them.
