PiCIE

Sunday, August 1, 2021 1:05 AM

https://sites.google.com/view/picie-cvpr2021/home

- Abstract
  - ⇒ A framework for semantic segmentation.
  - ⇒ Completely unsupervised: Inputs are only images. No labels.
  - ⇒ Classes are discovered by clustering of per-pixel features learned by deep networks
  - ⇒ Clustering pixel features alone does not work
    - Fail to learn high level concepts
    - Overfits to low-level visual cues
  - ⇒ Introduce inductive biases to make it all work out.
    1. Invariance to photometric transformations
    2. Equivaraance to geometric transformations
  - ⇒ Results
    1. Better results on COCO and Cityscapes datasets.
       - ⇒ Accuracy +17.5
       - ⇒ mIoU +4.5
    2. Better initialization for supervised learning
- Glossary
  - ⇒ Thing = foreground object
  - ⇒ Stuff = background object.
- Problem Statement
  - ⇒ Input = a set of unlabeled images in a domain D.
  - ⇒ Outputs
    - A set of visual classes C
    - A semantic segmentation function f

- A set of visual classes C
- A semantic segmentation function f_θ
  - ⇒ Given an image from D, assign to each pixel a class in C.

- Baseline clustering approach
  - ⇒ Based on DeepCluster https://arxiv.org/abs/1807.05520
    1. Start with an NN that outputs some pixel-level features
    2. Cluster with current features
    3. Use cluster labels as pseudo-labels to train a feature embedder
    4. Repeat 2 and 3 until satisfied.
  - ⇒ DeepCluster, however, only works at image level, not pixel level
  - ⇒ Let $\{x_1, x_2, \ldots, x_n\}$ be the set of training images.
  - ⇒ For each image x_i, $f_\theta(x_i)$ is a feature tensor containing embedding for each pixel.
  - ⇒ Let $f_\theta(x_i)[p]$ denote the embedding of pixel p of image x_i
  - ⇒ let $g_w(\cdot)$ = a classifier operating on the $f_\theta(x_i)[p]$'s.
  - ⇒ The baseline approach alternates between the following two steps.
    1. Use the current embedding to cluster the pixels by k-means

$$\min_{y,\mu} \sum_{i,p} \| f_\theta(x_i)[p] - \mu_{y_{ip}} \|^2$$

Here, y_{ip} = class label of $x_i[p]$.

μ_k = the mean vector associated with the k^{th} class.

k-means is computed with $\boxed{\text{minibatch k-means}}$

https://openreview.net/forum?id=B1EPCl-d-B

② Use the cluster labels to train a pixel classifier using standard cross entropy loss

$$\min_{\theta, w} \sum_{i,p} \mathcal{L}_{CE}\left(g_w(f_\theta(x_i)[p]),\ y_{ip} \right)$$

$$\mathcal{L}_{CE}\left((s_1, s_2, \dots, s_k),\ s_{y_{ip}} \right) = -\log \frac{e^{s_{y_{ip}}}}{\sum e^{s_k}}$$

Here, $(s_1, s_2, \dots, s_k) \leftarrow g_w(f_\theta(x_i)[p])$ is the class scores outputted by g_w.

- A modification to the baseline
  - ⇒ To prevent noisy gradients and noisy clusters, the paper removes the classifier g_w.
  - ⇒ Instead, it labels each pixels with the centroid of the nearest cluster.
  - ⇒ The score of a pixel embedding is the cosine distance from the embedding to the centroid.

$$\mathcal{L}_{clust}\left(f_\theta(x_i)[p],\ y_{ip},\ \mu \right) = -\log \left(\frac{e^{-d(f_\theta(x_i)[p],\ \mu_{y_{ip}})}}{\sum_\ell e^{-d(f_\theta(x_i)[p],\ \mu_\ell)}} \right)$$

$$d(v_1, v_2) = \text{cosine-distance}(v_1, v_2) = 1 - \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|}$$

- Inductive biases
  - ⇒ So far, there is no gaurantee that the features would be semantic.
  - ⇒ To get such features, the paper introduces two inductive biases.

① Invariance to photometric transformations : The labels should not

① Invariance to photometric transformations: The labels should not change if pixel colors are jittered slightly.

② Equivariance to geometric transformations: If the image is warped geometrically, then the labels should undergo the same transformation too.

Mathemathically: $G(P(x)) = G(Y)$

where x = input color image,

Y = image of labels

P = photometric transformation

G = geometric transformation

- Invariance to photometric transformation, operationally

⇒ For each image x_i, sample two random photometric transformations $P_i^{(1)}$, $P_i^{(2)}$.

⇒ This yields two feature embeddings:

$$z_{ip}^{(1)} = f_\theta(P_i^{(1)}(x_i))[p]$$

$$z_{ip}^{(2)} = f_\theta(P_i^{(2)}(x_i))[p]$$

⇒ We perform two k-means clustering separately, resulting in two labels and two centroids

$$y^{(1)}, \mu^{(1)} = \underset{y,\mu}{\operatorname{argmin}} \| z_{ip}^{(1)} - \mu_{y_{ip}} \|$$

$$y^{(2)}, \mu^{(2)} = \underset{y,\mu}{\operatorname{argmin}} \| z_{ip}^{(2)} - \mu_{y_{ip}} \|$$

⇒ To make sure that the features satisfy the clustering property, we minimize the clustering loss:

property, we minimize the following loss.

$$\mathcal{L}_{within} = \mathcal{L}_{clust}\left(f_\theta(P_i^{(1)}(x_i))[p],\ y_{ip}^{(1)},\ \mu^{(1)}\right)$$
$$+\ \mathcal{L}_{clust}\left(f_\theta(P_i^{(2)}(x_i))[p],\ y_{ip}^{(2)},\ \mu^{(2)}\right)$$

⇒ Because the labels should be invariant to photometric transformation, the following should also be minimized.

$$\mathcal{L}_{cross} = \mathcal{L}_{clust}\left(f_\theta(P_i^{(1)}(x_i))[p],\ y_{ip}^{(2)},\ \mu^{(2)}\right)$$
$$+\ \mathcal{L}_{clust}\left(f_\theta(P_i^{(2)}(x_i))[p],\ y_{ip}^{(1)},\ \mu^{(2)}\right)$$

⇒ The two losses, when minimize together, should encourage the network to learn an embedding function that

① produce features that will be labelled identically, and

② produce the same cluster centroids

regardless of the color transformations applied.

- Equivariance to geometric transformations, operationally.

⇒ In addition to the $P_i^{(1)}$ and $P_i^{(2)}$, we sample a random geometric transformation T_i for each image i.

⇒ Now, we compute

$$z_{ip}^{(1)} = f_\theta(G_i(P_i^{(1)}(x)))[p]$$
$$z_{ip}^{(2)} = G_i(f_\theta(P_i^{(2)}(x)))[p]$$

⇒ We then cluster and minimize $\mathcal{L}_{within} + \mathcal{L}_{cross}$ with the formula in the last section.

- Pseudocode

for $x_i \sim D$ do

$P_i^{(1)}, P_i^{(2)} \sim$ RandomPhotometricTransforms

$P_i^{(1)}, P_i^{(2)} \sim$ RandomPhotometricTransforms

$G_i \sim$ RandomGeometricTransform

$z_{i,:}^{(1)} \leftarrow G_i(f_\theta(P_i^{(1)}(x_i)))[:]$

$z_{i,:}^{(2)} \leftarrow f_\theta(G_i(P_i^{(2)}(x_i)))[:]$

end for

$\mu^{(1)}, y^{(1)} \leftarrow \text{KMeans}(\{z_{ip}^{(1)}\})$

$\mu^{(2)}, y^{(2)} \leftarrow \text{KMeans}(\{z_{ip}^{(2)}\})$

for $x_i \sim D$ do

$\mathcal{L}_{within} \leftarrow \sum_p \left(\mathcal{L}_{clust}(z_{ip}^{(1)}, y_{ip}^{(1)}, \mu^{(1)}) + \mathcal{L}_{clust}(z_{ip}^{(2)}, y_{ip}^{(2)}, \mu^{(2)}) \right)$

$\mathcal{L}_{cross} \leftarrow \sum_p \left(\mathcal{L}_{clust}(z_{ip}^{(1)}, y_{ip}^{(2)}, \mu^{(2)}) + \mathcal{L}_{clust}(z_{ip}^{(2)}, y_{ip}^{(1)}, \mu^{(1)}) \right)$

$\mathcal{L}_{total} \leftarrow \mathcal{L}_{within} + \mathcal{L}_{cross}$

$f_\theta \leftarrow \text{backward}(\mathcal{L}_{total})$

end

- Training details

⇒ The paper use Feature Pyramid Network https://arxiv.org/abs/1612.03144
with ResNet-18 backbone.

⇒ ResNet-18 is pretrained on ImageNet.

⇒ Fusion dimension of FPN = 128

⇒ L2 normalization on the feature map.

⇒ Images are resized and center cropped to 320 x 320.

⇒ Overclustering

- Previous works show that jointly optimizing for another higher cluster number leads to better stability and accuracy.

- The paper two cluster numbers together
  - ⇒ K_1 = one set by the user.
  - ⇒ K_2 is set to a fixed value of 100.
- The paper optimizes

$$\mathcal{L} = \frac{\log K_2}{\log K_1 + \log K_2} \mathcal{L}_{K_1} + \frac{\log K_1}{\log K_1 + \log K_2} \mathcal{L}_{K_2}$$

- Why these weights?
  - ⇒ Magnitude of cross entropy depends on $\log(K)$.

⇒ The paper says it also applies a balance term based on cluster size. However, it does not give enough details.

- Results can be found in the paper
  - ⇒ The new algo compares favorably to
    - Deep Cluster https://arxiv.org/abs/1807.05520
    - Invariant information clustering https://arxiv.org/abs/1807.06653