On the Effectiveness of Image Rotation for Open Set Domain Adaptation

Problem and background

- **Source Domain**: training set, labeled
- **Target Domain**: test set, unlabeled
- **OSDA**: Open-Set Domain Adaptation
- **Open-Set**: discrepancy between training data and test data, poor performances at test time. No guarantee that target domain has the same set of categories as the source domain, the source domain contains only a subset of the huge set of categories of the target domain. The model can learn only the categories of the source domain.
  We need to individuate and separate the known target samples from the unknown categories, rejecting the latter (**Unknown Detection** or **Anomaly Detection**)
  - **Openness**: the size of the source and target class set ($\mathbb{O} = 1 - \frac{|C_s|}{|C_t|}$)
  - In real-world problems the number of unknown target classes largely exceeds the number of known classes, with openness approaching to 1
- **Domain Adaptation**: perform the alignment, only between the source domain and the target-known domain, reducing the domain shift. We leverage a fully supervised data-rich source domain to learn a classification model that performs well on a different but related unlabeled target domain.
- Unknown Detection before Domain Adaptation avoid **negative transfer** (it occurs when the whole source and target distribution are forcefully matched, mistakenly aligning the unknown target samples with source data)
- **Self-supervised task**: training on pretext tasks (e.g., image colorization, rotation prediction), it uses unlabeled data to transfer the acquired high-level knowledge and domain-invariant regularities to new tasks with scares supervision. It applies the techniques of supervised learning on problems where external supervision is not available manipulating the data to generate the supervision for an artificial task that is helpful to learn useful feature representations
  It compensates for the lack of the not availability of supervised data or the expensiveness of annotating data
  - Used with Domain Adaptation, it learns robust cross-domain features and supports generalization, training a network to solve an auxiliary self-supervised task on the target (and source) data, in addition to the main task
  - Used with Anomaly Detection, it discriminates normal and anomalous data training a classifier on the normal data(Discriminative method)
  - **Rotation Recognition**: predicting the relative rotation (rotation angle) between a reference image and the rotated counterpart (rotating the input images by multiples of 90°), using a CNN as a classification task (label is 0°, 90°, 180°, 270°)

Proposed Solution

- **Rotation-based Open Set** (**ROS**): proposed two-stage method

- It abandons adversarial learning (of precedent OSDA methods) in favor of self-supervision and especially rotation recognition both to
  - Separate known and unknown target samples (Anomaly detection problem)
  - Align the known source and the target distributions (CSDA problem)

Unknown Detection

- **Rotation Recognition**: the network is trained on the Source Domain to predict the correct orientation of the objects
- The orientation is domain-invariant: if the network trained on the source-domain can predict the orientation of a target image, the object is known. Otherwise, the target image is discarded as unknown

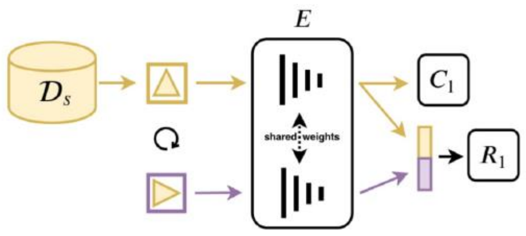
Domain Adaptation

- **Rotation Recognition** task in a multi-task approach, reducing the domain gap between the two distributions, leveraging on the generalized knowledge
- Using Standard Supervision

Implementation

Step1

- Source domain to train both **semantic object classifier $\boldsymbol{C_1}$** and **multi-rotation classifier $\boldsymbol{R_1}$**, with an **encoder $\boldsymbol{E}$**



- Semantic prefiction: $\hat{y}^s = softmax(C_1(E(x^s))$, only for regularization purposes and as a warm up for the following stage
- Rotation recognition task as a *classification problem* (angles of rotation 0°, 90°, 180°, 270° = classes 1, 2, 3, 4) with a CNN
  - To predict the image rotation, consider the *concatenation* of the features obtained with the not-rotated version (as anchor) and the rotated version of the same image
  - Network learns the ***relative rotation*** to avoid some ambiguity on absolute orientation, boosting discriminative power of the learned features and focusing more on specific shape details
  - To avoid suffering when multiple semantic categories are included in the normal class, the rotation reconition is extended (***multi-rotation classification***)
    - From a 4-class problem to a $(4 \times |C_s|)$-class problem, where the set of classes represents the combination of semantic and rotation labels
    - For example, if we rotate an image of category $y^s = 2$ by $i = 3$, its label for the multi-rotation classification task is $z^s = (y^s \times 4) + i = 11.$
  - Rotation prediction: $\hat{z}^S = softmax(R_1([E(x^s), E(\hat{x}^s)]))$

- Train two loss functions, one for the object recognition and one for the rotation recognition (multi-task learning), minimizing:

  $L_{tot} = L_{cls} + \alpha_1 L_{rot}$, where:
  - $\mathcal{L}_{C_1} = -\sum_{j \in \mathcal{D}_s} \boldsymbol{y}_j^s \cdot \log(\hat{\boldsymbol{y}}_j^s)$, defined as a *cross-entropy*
  - $\mathcal{L}_{R_1} = \sum_{j \in \tilde{\mathcal{D}}_s} -\lambda_{1,1} \boldsymbol{z}_j^s \cdot \log(\hat{\boldsymbol{z}}_j^s) + \lambda_{1,2} ||\boldsymbol{v}_j^s - \gamma(\boldsymbol{z}_j^s)||_2^2$, combining *cross-entropy* with *center loss* (encouraging the network to minimize the intra-class variations, keeping far the features of different classes and helping the detection of unknown category samples)
- If the rotation classifier R_1 can predict the orientation of a target image that means the image is from a known category
  - For each target image, **normality score** $\mathcal{N} \in [0,1]$ is the maximum prediction of R_1
    - Large $\mathcal{N}$ means normal (known) samples and vice-versa
    - Given the network prediction on all the relative rotations variants of a target sample $\hat{z}_i{}^t$ and their related entropy $H(\hat{z}_i{}^t) = \left(\hat{z}_i{}^t \cdot \frac{\log(\hat{z}_i{}^t)}{\log|C_s|}\right) with\ i = 1, \dots, |r|$
    - $\mathcal{N}(\boldsymbol{x}^t) = \max\left\{ \max_{k=1,\dots,|\mathcal{C}_s|} \left(\sum_{i=1}^{|r|} [\hat{\boldsymbol{z}}_i^t]_{k \times |r| + i} \right), \left(1 - \frac{1}{|r|} \sum_{i=1}^{|r|} H(\hat{\boldsymbol{z}}_i^t) \right) \right\}$, where we maximize over the *Rotation Score* (the ability of the network to correctly predict the semantic class and orientation of a target sample) and the *Entropy Score* (the confidence of the Rotation Score, evaluated on the basis of the predicion entropy), taking in each case the most reliable metric
    - Entropy helps to adapt with a more evident effect in case of large domain gaps

Algorithm 1 Compute normality score and Generate $\mathcal{D}_t^{knw}$ & $\mathcal{D}_t^{unk}$

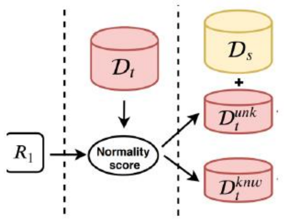
```
Input:
  Trained networks $E$ and $R_1$
  Target dataset $\mathcal{D}_t = \{x_j^t\}_{j=1}^{N_t}$
Output:
  Known target dataset $\mathcal{D}_t^{knw} = \{x_j^{t,knw}\}_{j=1}^{N_{t,knw}}$
  Unknown target dataset $\mathcal{D}_t^{unk} = \{x_j^{t,unk}\}_{j=1}^{N_{t,unk}}$

  procedure GETROTATIONSCORE($z$,$i$)
    $\mathbf{o} = zeros(|\mathcal{C}_s|)$ # vector of $|\mathcal{C}_s|$ zeros
    for each $k$ in $\{1, ..., |\mathcal{C}_s|\}$ do
      $[\mathbf{o}]_k = [z]_{k \times 4 + i}$ # $[\mathbf{a}]_b$ indicated the $b$-th element of vector a
    return o
  procedure GETENTROPYSCORE($z$)
    return $z \cdot \log(z) / \log(|\mathcal{C}_s|)$
  procedure GETNORMALITYSCORE($E$,$R_1$,$\mathcal{D}_t$)
    for each $x_j^t$ in $\mathcal{D}_t$ do
      Initialize: $h = \{\}$, $\mathbf{o} = zeros(|\mathcal{C}_s|)$
      for each $i$ in $\{1, ..., 4\}$ do
        $\tilde{x}_j = rot90(x_j, i)$
        $z_j = \text{softmax}(R_1(E(x_j)||E(\tilde{x}_j)))$
        $h \leftarrow getEntropyScore(z_j)$
        $o += getRotationScore(z_j, i)$ # element-wise sum of vectors
      $h = mean(h)$
      $o = max(\mathbf{o})$
      $\mathcal{N} \leftarrow \eta_j = max(o, 1 - h)$
    return $\mathcal{N}$

  procedure MAIN( )
    Initialize: $\mathcal{D}_t^{knw} = \{\}$, $\mathcal{D}_t^{unk} = \{\}$
    $\mathcal{A} = getNormalityScore(E, R_1, \mathcal{D}_t)$
    for each $(x_j, \eta_j)$ in $(\mathcal{D}_t, \mathcal{N})$ do
      if $\eta_j \geq mean(\mathcal{N})$ then
        $\mathcal{D}_t^{knw} \leftarrow x_j$
      else
        $\mathcal{D}_t^{unk} \leftarrow x_j$
```

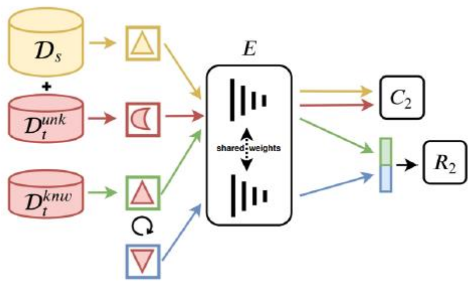
- Evaluate the precision with the **AUROC** (*Area Under Receiver Operating Characteristic curve*) metric over the normality score $\mathcal{N}$ (the method is working if the value > 0.5)
- Known-unknown separation of the target, setting a threshold value (if the *normality score* > threshold, the target sample is considered known, otherwise unknown)
  - We use as threshold the average of the normality score over the whole target $\bar{\mathcal{N}} = \frac{1}{\mathcal{N}_t} \sum_{j=1}^{\mathcal{N}_t} \mathcal{N}_j$

$$\begin{cases} \boldsymbol{x}^t \in \mathcal{D}_t^{knw} & \text{if} \quad \mathcal{N}(\boldsymbol{x}^t) > \bar{\mathcal{N}} \\ \boldsymbol{x}^t \in \mathcal{D}_t^{unk} & \text{if} \quad \mathcal{N}(\boldsymbol{x}^t) < \bar{\mathcal{N}} \end{cases}$$



Step2

- Unknown part of the target used for training the unknown class (with the semantic classifier C_2), extending the original semantic classifier and making it able to recognize the unknown category
- Known part of the target used for the source-target adaptation with the rotation classifier R_2, without the risk of negative transfer



- The encoder E is inherited from the previous stage
- C_2 is similar to C_1 but it has a $(|C_s|+1)$ - *dimensional output*, having in addition the unknown class
  - Semantic prediction: $\hat{g} = softmax(C_2(E(x))$
- R_2 is similar to R_1 but it is a **simple rotation classifier with a 4-dimensional output**
  - Rotation prediction: $\hat{q} = softmax(R_2([E(x), E(\hat{x})]))$
- Multi-task learning, minimizing the objective function:

$$L_{tot} = L_{cls} + \alpha_2 L_{rot}$$

, where:

- $$\mathcal{L}_{C_2} = - \sum_{j \in \{\mathcal{D}_s \cup \mathcal{D}_t^{unk}\}} \boldsymbol{g}_j \cdot \log(\hat{\boldsymbol{g}}_j) - \lambda_{2,1} \sum_{j \in \mathcal{D}_t^{knw}} \hat{\boldsymbol{g}}_j \cdot \log(\hat{\boldsymbol{g}}_j)$$, combining a supervised cross-entropy and an unsupervised entropy loss (used to involve the unlabeled target samples recognized as known, enforcing the decision boundary to pass through low-density areas)

- $\mathcal{L}_{R_2} = -\lambda_{2,2} \sum_{j \in \mathcal{D}_t^{knw}} \boldsymbol{q}_j \cdot \log(\hat{\boldsymbol{q}}_j)$, defined as a cross-entropy loss
- Once training is complete, R_2 is discarded and the target labels are predicted as:
  - $c_j^t = C_2(E(\boldsymbol{x}_j^t))$ for all $j = 1, \ldots, N_t$

Step3

- Evaluate the final model on the target domain according to the metrics
  - OS* → accuracy of the object classifier to recognize the known category
    -
  - UNK → accuracy of the object classifier to recognize the unknown category
  - **HOS** → harmonic mean between OS* and UNK
    - $2 \frac{\mathrm{OS}^* \times UNK}{\mathrm{OS}^* + UNK}$
    - Past metrics have the drawbacks of the vanishing of the contribution of the unknown classes with increasing number of known classes
    - This metric properly balances the contribution of recognizing the known classes and rejecting the unknown samples
    - It provides a high score only if the algorithm performs well both on known and unknown samples, independently $|C_s|$, so only if both OS* and UNNK are high
    - It penalizes large gaps between OS* and UNK

Step4

- Change the values of α_1, α_2 and threshold in a range of possible values, seeing what happens to the performances of the model

Experiments

Datasets

- **Office-31**:
  - Three domains - Webcam (W), Amazon (A), Dsrl (D)
  - The first 10 classes in alphabetic order are considered known
  - The last 11 classes are considered unknown
- **Office-Home**
  - Four domains – Product (Pr), Art (Ar), Real World (Rw), Clipart (Cl)
  - The first 25 classes in alphabetic order are considered known classes
  - The remaining 40 classes are considered unknown

Implementation Details

- **ResNet-50** and **VGGNet** backbone implementation
  - Pretrained on ImageNet
  - Batch size: 32
  - Learning Rate: 0.0003, inverse decay scheduling
    - On the layers trained from scratch, LR is 10 times higher than the pre-trained ones
  - SGD
    - Weight decay: 0.0005
    - Momentum: 0.9

    - $\lambda_{1,1} = \lambda_{2,2} = 3$
    - $\lambda_{1,2} = \lambda_{2,1} = 0.1$
    - Each experiment is repeated three times taking the result on the target at the last epoch
- **ResNet-50**
    - Encoder E
        - All the layers of a standard ResNet-50 up to the *average pooling layer*
        - Pre-trained on ImageNet, updating only the last *convolutional block* (finetuning with LR=0.0003)
    - Classifiers C_1, C_2: two *Fully Connected* (FC) layers
        - The first FC has output 256, followed by a *Batch Normalization* layer and a *Leakly-ReLU* (negative slope angle as 0.2)
        - The second FC
            - For C_1: $|C_s|$ outputs
            - For C_2: $|C_s| + 1$ outputs (including the unknown category)
        - All layers are learned from scratch (LR=0.0003)
    - Rotation classifiers R_1, R_2: same structure of classifiers C_1, C_2
        - For R_1: $4 \times |C_s|$ outputs
        - For R_2: 4
        - All layers are learned from scratch (LR=0.0003)
    - Network trained in Stage I used as starting point for Stage II
        - In Stage II, LR of the new unknown class is 2 times higher than the known classes already learned in Stage I
    - For *Office-31*, 80 epochs training for Stage I and 80 for Stage II
    - Only for *Office-Home*, $\lambda_{1,2} = 0.001$
    - For *Office-Home*, 150 epochs training for Stage I and 45 for Stage II
- **VGGNet**
    - Encoder E
        - All the layers of a standard VGG-19 up to the *fully connected layer*
        - Pre-trained on ImageNet, updating only the last 2 *FC* (finetuning with LR=0.0003)
    - Classifiers C_1, C_2, R_1, R_2: same structure of ResNet-50 implementation
    - Network trained in Stage I not used as starting point for Stage II
        - In Stage II, LR of the new unknown class is 1.5 times higher than the known classes already learned in Stage I
    - For *Office-31*, 100 epochs training for Stage I and 200 for Stage II

Results

- ROS outperforms its competitors, defining a new state-of-art on two benchmark datasets
- Great ability in separating known and unknown samples, differently from the competing methods
- It generalizes across datasets and network architectures without specific finetuning of the hyper-parameters
- **Reproducibility** is a crucial issue nowadays in Machine Learning, urgent need of a rigorous experimental validation, being consistent between results on the paper and on the code
    - For this reason in this paper they re-run the state-of-art methods for ODSA, comparing the results reported in the papers with the results produced using the original public implementation and their parameters

    - They suggest to not simply copy the results reported in the papers
- In contrast to the other methods, ROS maintains a consistent steady performance also increasing the openness
- ROS is even twice as fast as its best competitor in term of HOS performance
- HOS metric is a reliable tool to allow fair open set evaluation

GitHub Repo

- https://github.com/silvia1993/ROS