भारतीय सांकेतिक विज्ञान संस्थ हैदराबाद
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Task-oriented Alignment for Unsupervised Domain Adaptation

Presented at **NeurIPS 2021**

CS6450 : Visual Computing

Presentation - 2

Supervisor:

Prof. C. Krishna Mohan

Presenter :
Shrusti
CS22MTECH11017

Teaching Assistant :
Aveen Dayal

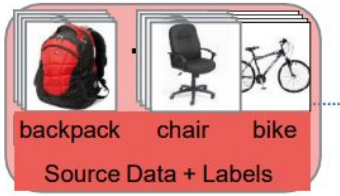
Contents

Motivation

- The trained models typically perform well when applied to testing data, which has same data distribution to the training data.

- However, in practice we observe performance degradation due to **Domain shift.**

Myth : You cant do deep learning unless you have million labelled examples for your problem.

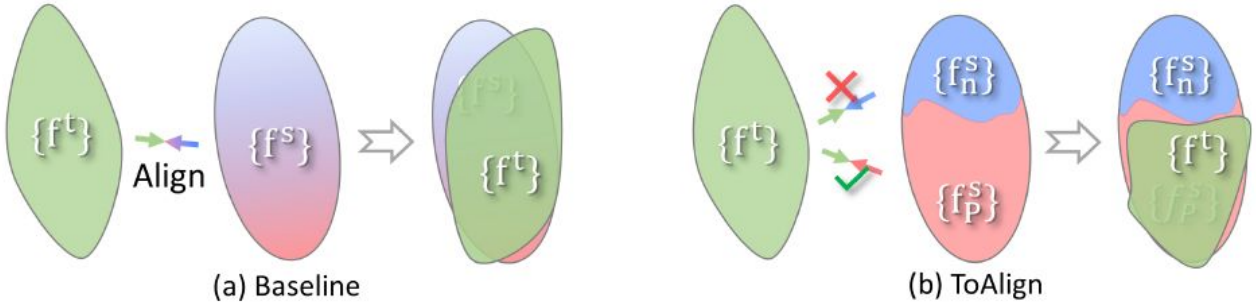


Reality :

1) You can learn useful representations from unlabelled data
2) You can transfer learned representations from a related task

Labelled data is available sometimes in cases of image recognition, speech recognition, recommendation, etc
But difficult to collect sometimes in Robotics, Disaster, Medical diagnosis, etc

Paper's Method : Task Oriented Alignment



- We decompose a source sample feature into a task-discriminative one that should be aligned, and a task-irrelevant one that should be ignored under the guidance of classification-meta knowledge for performing **classification-oriented alignment**, which explicitly guides the network what features should be aligned.

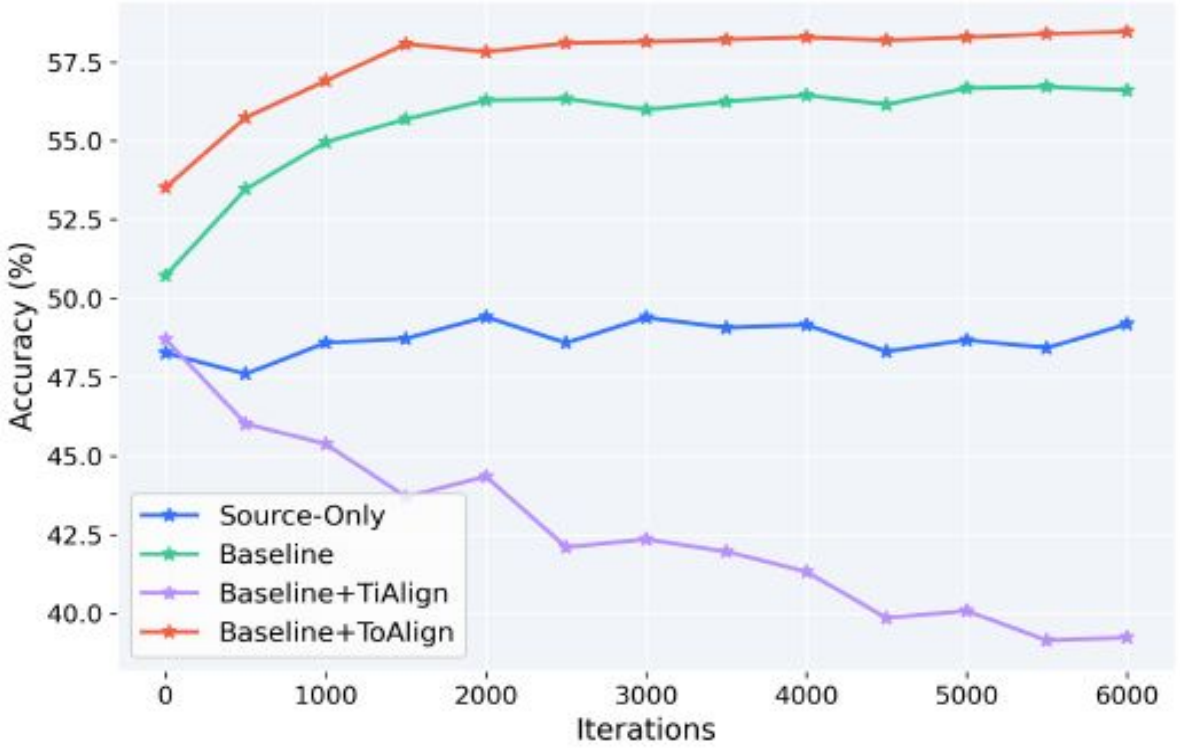
- Then, we perform alignment between the target features and the positive source features, which is consistent with the essence of the classification task.

- **Domain invariant features** - are those features that are consistent between the source and the target domain.

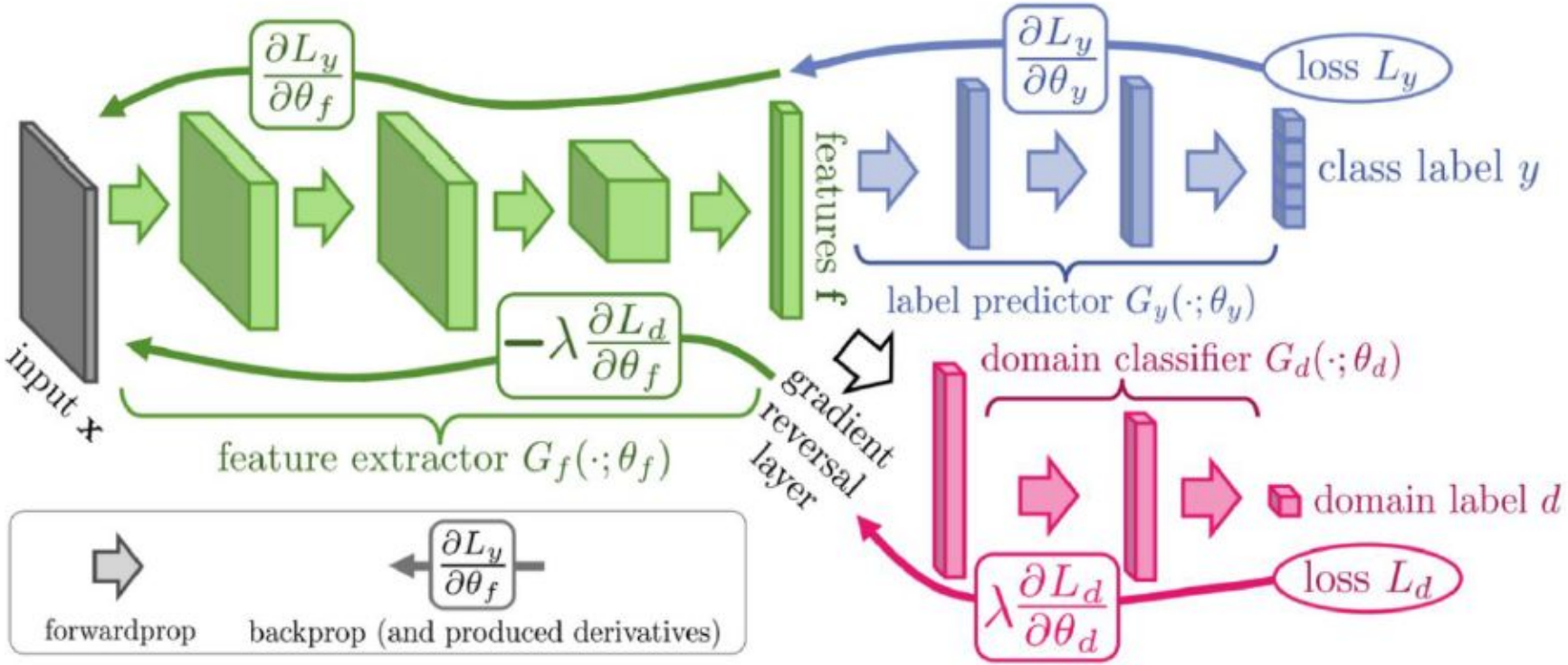
➢ And our goal is to learn domain invariant features.

Task oriented feature decomposition

Mistakenly aligning the target features with the source task-irrelevant features would hurt the discrimination power of the target features.



Domain Adversarial UDA



Label predictor : This is a neural network that will learn to **perform classification** on the transformed source distribution. Since, source domain is labelled.

Domain classifier : This is a neural network that predicts whether output from feature extractor is from **source domain or target domain**.

- In order to identify which domain a sample belongs to, domain adversarial learning-based UDAs typically train a domain discriminator D, and then adversarially train a feature extractor G to fool the discriminator D in order to learn domain-invariant feature representations.

Intuition : Feature extractor will try to perform some transformation on source and target instances such that the transformed instances appear as it is coming from same domain.

- Particularly, D is optimised to **minimise domain classification** loss L_D and G is optimised to **maximise domain classification loss** L_D and **minimise image classification loss** L_{cls}.

$$\underset{D}{\operatorname{argmin}} \mathcal{L}_D,$$
$$\underset{G}{\operatorname{argmin}} \mathcal{L}_{cls} - \mathcal{L}_D,$$

- L_D is typically defined as (BCE loss)

$$\mathcal{L}_D(\mathbf{X}_s, \mathbf{X}_t) = -\mathbb{E}_{\mathbf{x}_s \sim \mathbf{X}_s}\left[\log(D(G(\mathbf{x}_s)))\right] - \mathbb{E}_{\mathbf{x}_t \sim \mathbf{X}_t}\left[\log(1 - D(G(\mathbf{x}_t)))\right]$$

And L_{cls} is the Cross Entropy loss.

- **<u>Gradient Reversal Layer</u>** : is placed between feature extractor and domain classifier, it helps to maximise the domain classification loss .

It basically acts as an identity function(output is same as input) during forward propagation and during backpropagation it multiplies input by -1 leading to the opposite of gradient descent.

DANNP

ToAlign is applied on top of two different baselines schemes **DANNP and HDA**.

DANNP : DANNP is an improved version of DANN (Domain Adversarial Neural Network). It is different from DANN in two ways:

1) the input to the discriminator (D) is the predicted classification probability.

2) the samples that are easier to transfer(low entropy i.e; the model is more confident and certain about the class labels of these samples) are prioritized.

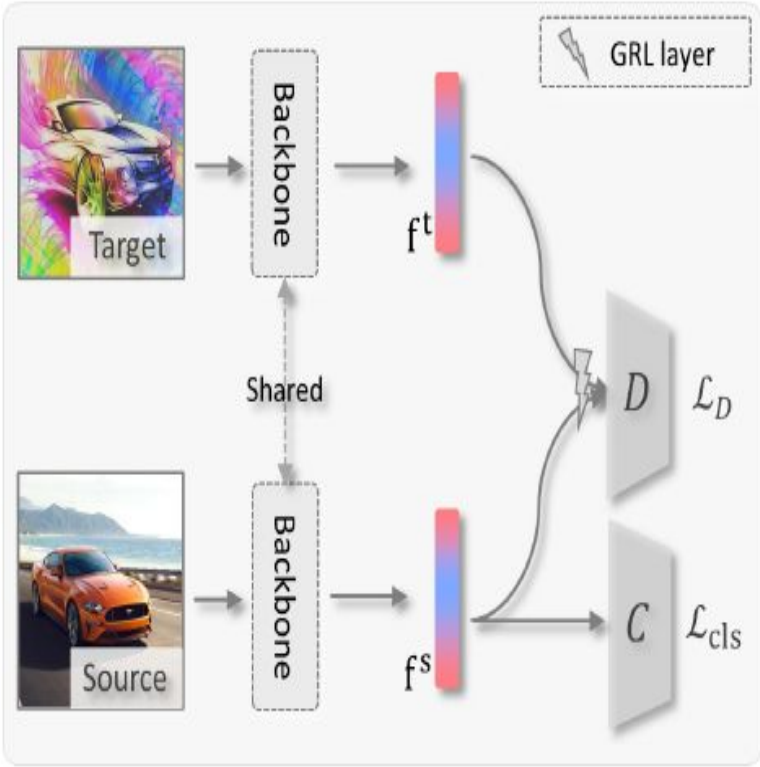
HDA : separates domain-specific characteristics from domain-invariant representations.

- They do this by using a process called **heuristic search.**

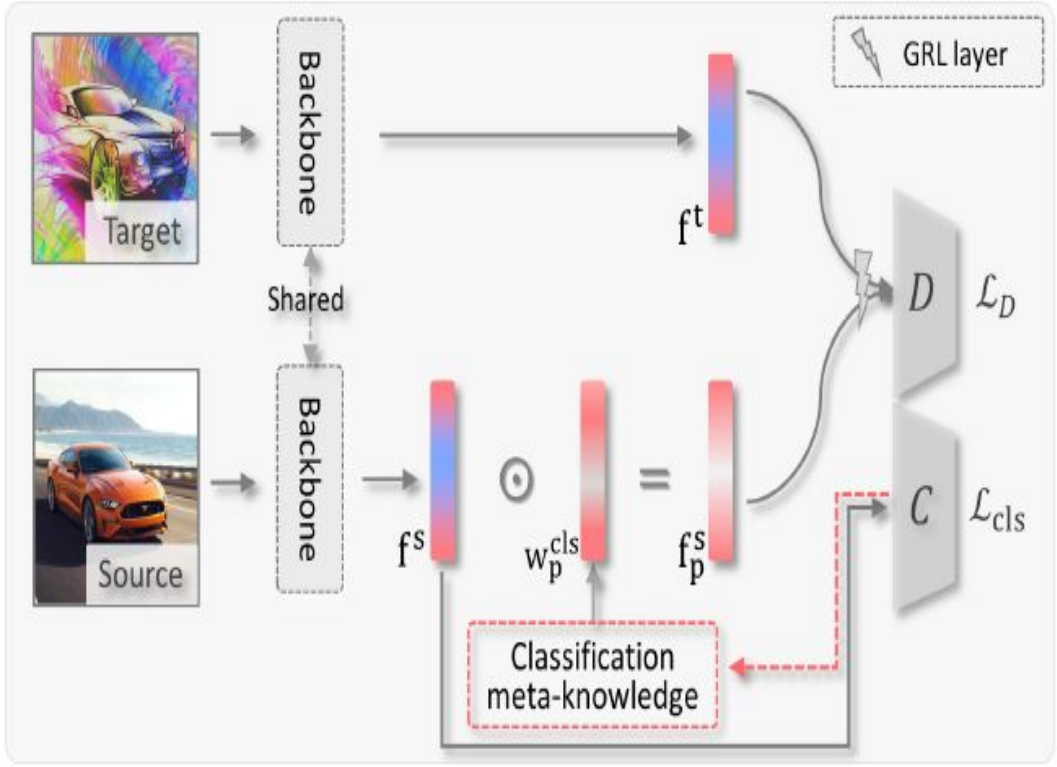
- Heuristic search is a problem solving approach that makes **educative guesses** to search for a solution in problem space.

- Heuristics might include features of the input data, data preprocessing techniques, adjusting hyperparameter or modifications to the model architecture or training process.

- idea is to find the solution by making informed guesses based on the problem's characteristics, rather than trying all possible combinations of options.



(a) Baseline



(b) ToAlign

Experiments

To evaluate effectiveness of ToAlign we conducted experiment under **SUDA** (Single source Unsupervised Domain Adaptation).

SUDA - training a model on a **single source domain**, and then adapting it to perform well on a target domain, without the need for labeled data in the target domain

Baselines used are

1. **DANNP** is an improved version of DANN where only **privileged information** (positive source features) are used
2. **HDA**

Datasets :

Office - Home

a) Consists of images from four different domains: **Art (Ar), Clipart (Cl), Product (Pr),and Real-World (Rw).**
b) Each domain contains **65** object categories in office and home environments.

Office - Home dataset

*Sample images from the **Office-Home** dataset. The dataset consists of images of everyday objects organized into 4 domains; **Art**: paintings, sketches and*or artistic depictions, **Clipart**: clipart images, **Product**: images without background and **Real-World**: regular images captured with a camera.

Architecture

- **Backbone**(starting point) : ResNet 50
- **Image classifier** : One fully connected
- The discriminator D consists of **three fully connected layers** with inserted dropout and **ReLU** layers.
    - Dropout means during training certain activations in the network are dropped which helps to prevent **overfitting** of network.
- Domain Adaptation Setting : Single Source Unsupervised Domain Adaptation

- **Learning rate** is set as

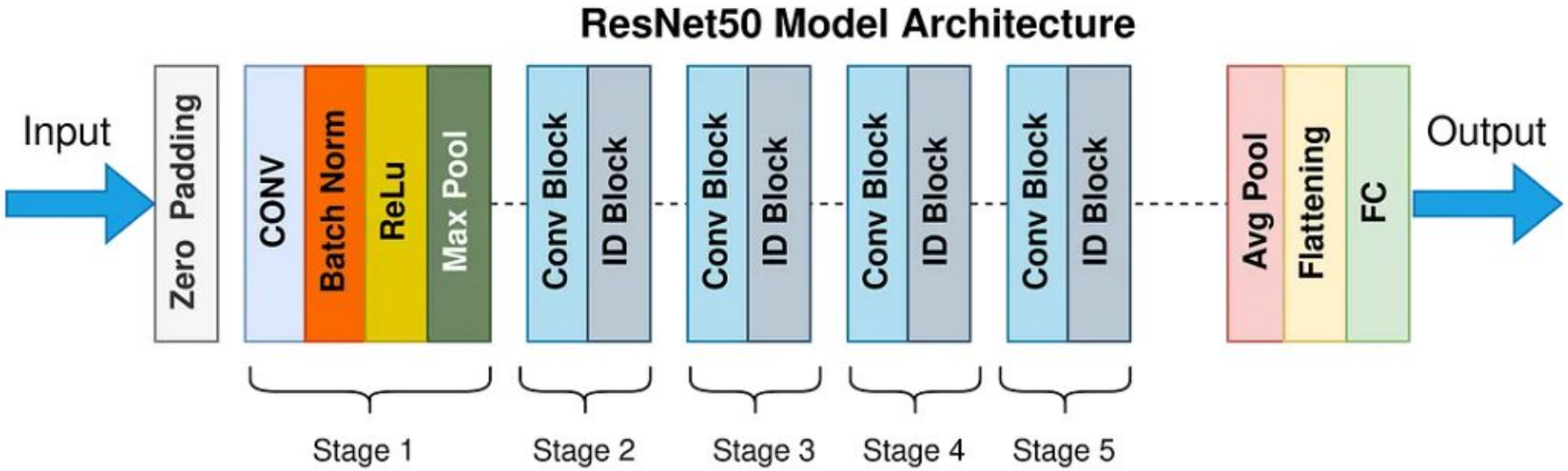
$$\eta_t = \frac{\eta_0}{(1+\gamma p)^\tau}$$

  - where **p** indicates the **progress** of training that increases linearly from 0 to 1, it controls how quickly or slowly the learning rate should change.
  - γ indicates the **decay rate** (i.e; gradually reduce the learning rate overtime training process helping the model to converge towards better optimum), it is a regularization technique that helps prevent overfitting
  - τ controls the strength of the regularization applied.

- Initial values are : **γ = 5e-4, τ = 0.75, η_0 = 1e - 3** for Office-home.

- Batch size : 36
- Optimiser : SGD with a momentum of 0.9

ResNet 50


ResNet50 Model Architecture
Input
Zero Padding
CONV
Batch Norm
ReLu
Max Pool
Conv Block
ID Block
Conv Block
ID Block
Conv Block
ID Block
Conv Block
ID Block
Avg Pool
Flattening
FC
Output
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5

Implementation Details

Solution : Reweight the feature vector f{s} to get f{s,p} and f{s, n}, this is done using Grad CAM.

Grad CAM identifies the relevant features to recognise the image correctly.

It generate a heatmap that highlights the most important regions in the image for the class prediction.

We obtain a feature map F with height H, width W, and M channels from the final convolutional block of the feature extractor. After spatial-wise global average pooling (GAP), we have a feature vector

$$\mathbf{f = pool(F) \text{ belongs to } R^M}$$

GAP is a type of pooling that computes **average of each channel** of feature map over height and width.

- The labels of the class are predicted via classifier C(.), based on response of classifier we derive the gradient w_{cls} of y^k w.r.t. f
- Gradient captures the influence of each dimension of the feature on prediction class k and these gradients highlights the regions of input that are most important for prediction of class k.

$$\mathbf{w}^{cls} = \frac{\partial y^k}{\partial \mathbf{f}}$$

where y^k is the predicted score corresponding to the ground-truth class k.

- Grad CAM uses w^{cls} to obtain discriminative features(i.e positive features) as :

$$\mathbf{f}_p = \mathbf{w}_p^{cls} \odot \mathbf{f} = s\mathbf{w}^{cls} \odot \mathbf{f},$$

Where s Є R_+ is a non-negative parameter called as **'attention score'** which gives higher weights to the elements that are important for the task and lower weights to the elements that are less important.

- This helps in identifying task-relevant features and suppresses the task-irrelevant features.

$$s = \sqrt{\frac{||\mathbf{f}||_2^2}{||\mathbf{w}^{cls} \odot \mathbf{f}||_2^2}}$$

- The attention score s is computed as the ratio of the squared L2 norm (because it measures the strength of the feature vector) of the feature map to the squared L2 norm of the weighted feature map, which measures the **degree of contribution of each channel to the weighted feature map.**

```
def _forward_impl(self, x):
    x = self.conv1(x)
    x = self.bn1(x)
    x = self.relu(x)
    x = self.maxpool(x)
    x = self.layer1(x)
    x = self.layer2(x)
    x = self.layer3(x)
    x = self.layer4(x)
    return x

def forward_backbone(self, x):
    x = self._forward_impl(x)
    x = self.global_avgpool(x)
    f = torch.flatten(x, 1)
    return f
```

f = pool(F) belongs to R^M

```
if toalign:
    w_pos = self._get_toalign_weight(f, labels=labels)
    f_pos = f * w_pos
    y_pos = self.fc(f_pos)
```

$$\mathbf{f}_p = \mathbf{w}_p^{cls} \odot \mathbf{f} = s\mathbf{w}^{cls} \odot \mathbf{f},$$

```
def _get_toalign_weight(self, f, labels=None):
    assert labels is not None, f'labels should be asigned'
    w = self.fc.weight[labels].detach()  # [B, C]
    if self.hda:
        w0 = self.fc0.weight[labels].detach()
        w1 = self.fc1.weight[labels].detach()
        w2 = self.fc2.weight[labels].detach()
        w = w - (w0 + w1 + w2)
    eng_org = (f**2).sum(dim=1, keepdim=True)  # [B, 1]
    eng_aft = ((f*w)**2).sum(dim=1, keepdim=True)  # [B, 1]
    scalar = (eng_org / eng_aft).sqrt()
    w_pos = w * scalar

    return w_pos
```

$$s = \sqrt{\frac{||\mathbf{f}||_2^2}{||\mathbf{w}^{cls} \odot \mathbf{f}||_2^2}}$$

Results

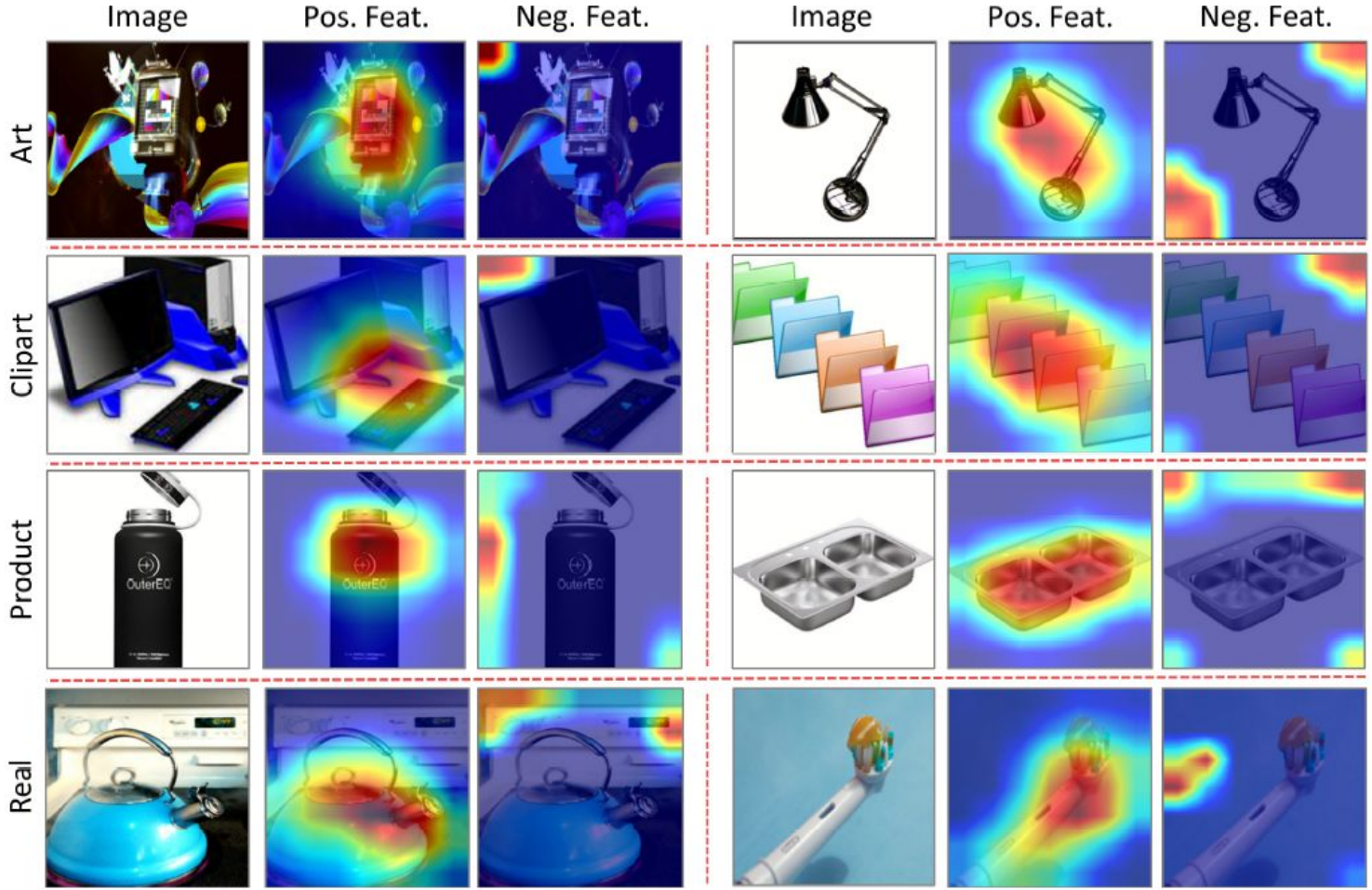


Image
Pos. Feat.
Neg. Feat.
Image
Pos. Feat.
Neg. Feat.
Art
Clipart
Product
Real

	Paper's Accuracies in %	Reproduced Accuracies in %
Art -> ClipArt	57.9	57.2
Art -> Product	76.9	76.3
Art -> Real World	80.8	77.9
ClipArt -> Art	66.7	59.7
ClipArt -> Product	75.6	73.5
ClipArt -> Real World	77.0	71.3
Product -> Art	67.8	62.4
Product -> ClipArt	57.0	56.5
Product -> Real World	82.5	79.8
Real World -> Art	75.1	74.2
Real World -> ClipArt	60.0	59.2
Real World -> Product	84.9	84

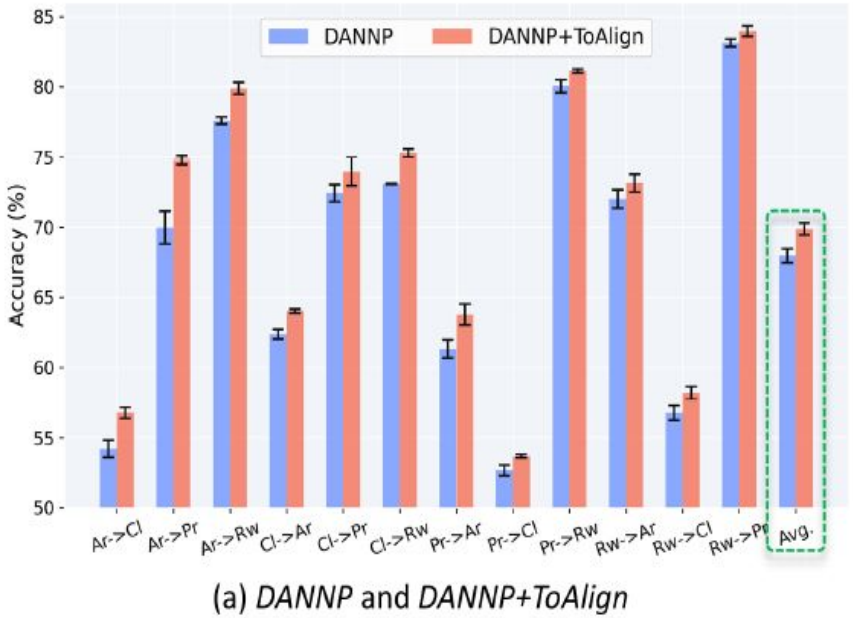
Batch size used in paper : 36

Batch size I used : 15

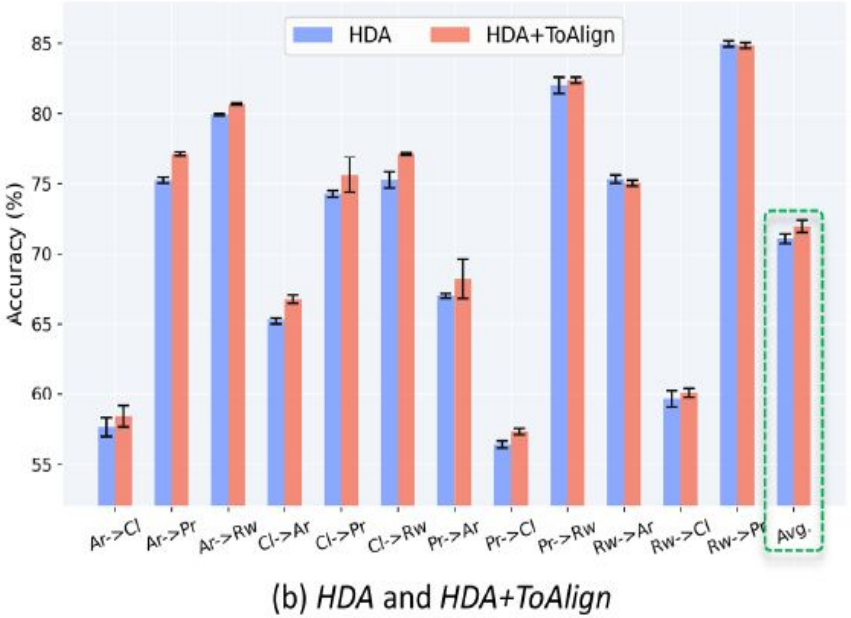
Reason :

1) Convergence : as batch size is low, it may need more number of iterations to converge

2) Randomness



(a) *DANNP* and *DANNP+ToAlign*



(b) *HDA* and *HDA+ToAlign*

Error bars of ToAlign on top of DANNP and HDA on Office-Home

Results obtained : Art ->Product


```
[2023-03-28 10:17:57] ======================= args =======================
{
    "cfg": "configs/uda_office_home_toalign.yaml",
    "seed": 123,
    "source": [
        "a"
    ],
    "target": [
        "p"
    ],
    "output_root": "/content/drive/MyDrive/VisCom/UDA-main/output",
    "output_dir": null,
    "data_root": "/content/drive/MyDrive/VisCom/UDA-main/dataset_map/office_home_dataset",
    "opts": null
}
[2023-03-28 10:17:57] ======================= cfg =======================
DATASET:
    NAME: office_home_dataset
    NUM_CLASSES: 65
    ROOT: /content/drive/MyDrive/VisCom/UDA-main/dataset_map/office_home_dataset
    SOURCE:
    - art
    TARGET:
    - product
    TRIM: 0
METHOD:
    ENT: true
    HDA:
        LR_MULT: 1.0
        W_HDA: 1.0
    W_ALG: 1.0
MODEL:
    BASENET: resnet50
    BASENET_DOMAIN_EBD: false
```

This is the Log file for Art->Product domains where Art is the source domain and Product is the target domain.

```
[2023-03-28 13:37:39] --> testing on source_test
[2023-03-28 13:38:09]     I:  49/68 | acc: 99.546
[2023-03-28 13:38:16] --> testing on target_test
[2023-03-28 13:38:32]     I:  49/124 | acc: 81.179
[2023-03-28 13:38:44]     I:  99/124 | acc: 81.818
[2023-03-28 13:38:50] I:  7999/8000 | src_acc: 99.629 | tar_acc: 76.301 | best_acc: 76.346
[2023-03-28 13:38:51]     models saved to /content/drive/MyDrive/VisCom/UDA-main/output/ckpt/art2product_123/models-last.pt
[2023-03-28 13:38:51]     keys: dict_keys(['optimizer', 'ite', 'best_acc', 'base_net', 'd_net'])
```

Results obtained : ClipArt ->Art


```
[2023-03-29 07:58:57] ======================= args =======================
{
    "cfg": "configs/uda_office_home_toalign.yaml",
    "seed": 123,
    "source": [
        "c"
    ],
    "target": [
        "a"
    ],
    "output_root": "output",
    "output_dir": null,
    "data_root": "dataset_map/office_home_dataset",
    "opts": null
}
[2023-03-29 07:58:57] ======================= cfg =======================
DATASET:
    NAME: office_home_dataset
    NUM_CLASSES: 65
    ROOT: dataset_map/office_home_dataset
    SOURCE:
    - clipart
    TARGET:
    - art
    TRIM: 0
METHOD:
    ENT: true
    HDA:
        LR_MULT: 1.0
        W_HDA: 1.0
    W_ALG: 1.0
MODEL:
    BASENET: resnet50
    BASENET_DOMAIN_EBD: false
```

This is the Log file for ClipArt->Art domains where ClipArt is the source domain and Art is the target domain.

```
[2023-03-29 08:49:14] --> testing on source_test
[2023-03-29 08:49:18]     I:  57/291 | acc: 99.532
[2023-03-29 08:49:22]     I:  115/291 | acc: 98.609
[2023-03-29 08:49:27]     I:  173/291 | acc: 98.150
[2023-03-29 08:49:31]     I:  231/291 | acc: 98.499
[2023-03-29 08:49:35]     I:  289/291 | acc: 98.385
[2023-03-29 08:49:35] --> testing on target_test
[2023-03-29 08:49:39]     I:  49/162 | acc: 67.891
[2023-03-29 08:49:43]     I:  99/162 | acc: 65.051
[2023-03-29 08:49:47]     I:  149/162 | acc: 59.911
[2023-03-29 08:49:48] I:  7999/8000 | src_acc: 98.396 | tar_acc: 59.168 | best_acc: 59.662
[2023-03-29 08:49:49]     models saved to output/ckpt/clipart2art_123/models-last.pt
[2023-03-29 08:49:49]     keys: dict_keys(['optimizer', 'ite', 'best_acc', 'base_net', 'd_net'])
```

Limitations

1) **Lack of interpretability:**
   a) Task-oriented alignment is a data-driven approach that relies on the similarity between the feature representations of the source and target domains. As such, it can be difficult to interpret the alignment process and understand how the adapted model is making predictions.

2) **Limited domain shift:**
   a) To Align assumes that the domain shift between the source and target domains is limited and can be captured by a linear transformation. However, in practice, domain shifts can be much more complex .

3) This **lack of understanding** of the knowledge transfer in DA models might raise doubts on their real-life applications especially when the stakes are high.

Future Work

1) Instead of using Grad CAM we can try with **Grad CAM ++.**

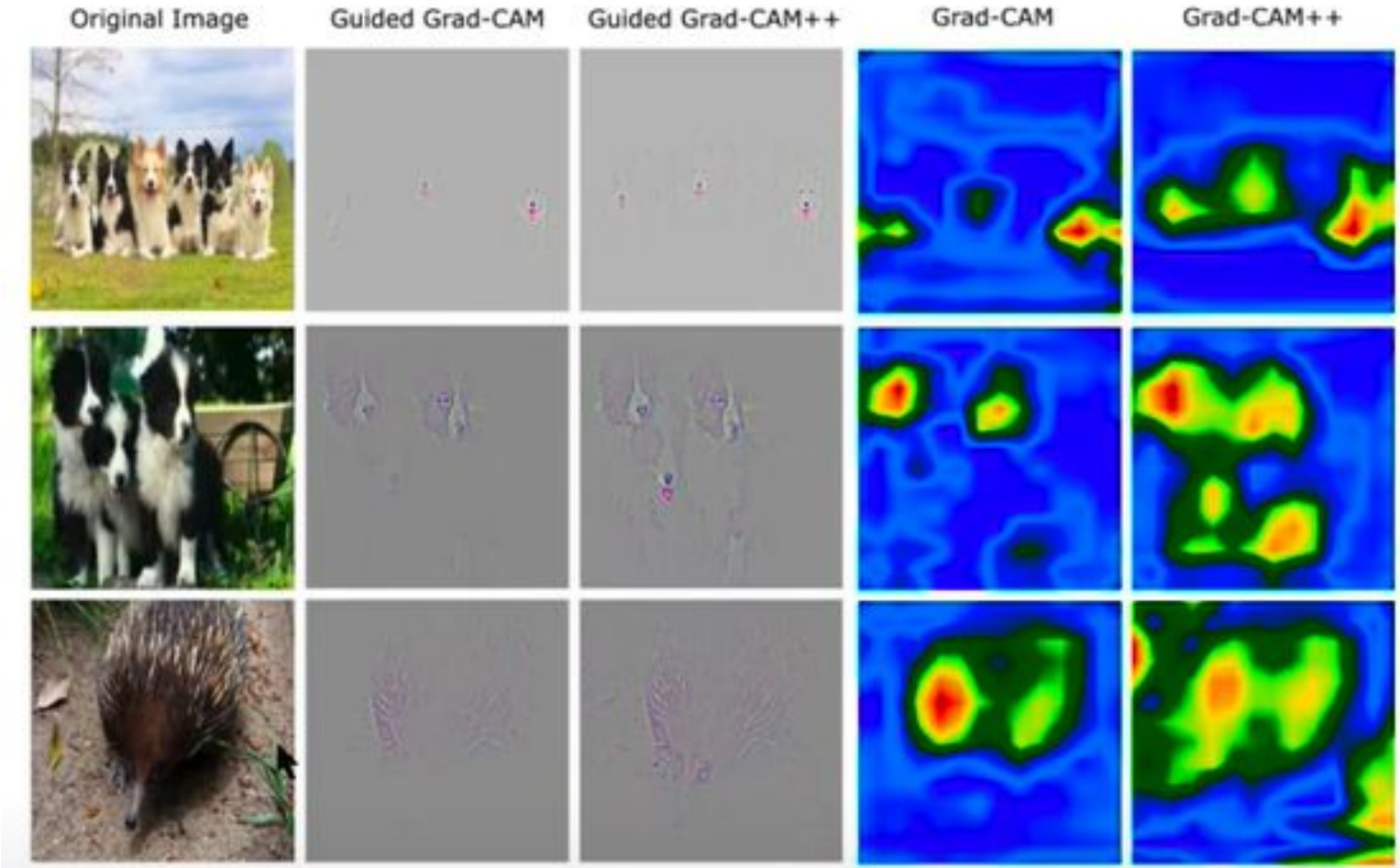
   Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks

   https://arxiv.org/pdf/1710.11063.pdf

- Not able to identify multiple instances of objects.

- Suppresses activation maps which have lesser spatial footprint.

- When computing weights of activation maps Grad CAM considers all pixel weights equally.


Original Image
Guided Grad-CAM
Guided Grad-CAM++
Grad-CAM
Grad-CAM++

- Whereas Grad CAM ++ uses ‘pixel-wise weight’ i.e; **α** . This pixel-wise weight tells how they must contribute to the saliency maps.

$$w_k^c = \sum_i \sum_j \alpha_{ij}^{kc} \quad \underbrace{ReLU\left(\frac{\partial y^c}{\partial A_{ij}^k}\right)}$$

$$\alpha_{ij}^{kc} = \frac{\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2}}{2\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2} + \sum_a \sum_b A_{ab}^k \{\frac{\partial^3 Y^c}{(\partial A_{ij}^k)^3}\}}$$

- The value depends on the second and third partial derivatives of the loss function with respect to activation map, as well as the values of it at all other pixel locations. It can be calculated using these values during training, and is used to adjust the gradient computed during backpropagation so that the model parameters are updated more effectively.

2) **Visualising the knowledge transfer** of Unsupervised Domain Adaptation

https://arxiv.org/pdf/2303.02302v1.pdf

"*This target image part looks like that source sample part since they share the same semantics*." Overall, the proposed model aims to provide a visual interpretation of the transferred knowledge in unsupervised domain adaptation.

Consists of 3 models in it :

a) **Knowledge extraction** : Extracts the features learnt from the pre-trained model.

It helps to identify the specific feature which is used to distinguish it from others.

b) **Prediction calibration** : new model learns to predict based on how similar the new data is to the extracted features

Considers the original model's predictions on the same data because it has already learned to recognize patterns that are relevant to the task.

c) **Knowledge fidelity preservation**: This model ensures that the extracted knowledge is accurate and faithfully represents the original information learned by the pre-trained model.

It does this by comparing the output probabilities of the new model and the original pre-trained model on the same data, and then making sure that the distribution of predicted probabilities is aligned between the two models.

Essentially, this model checks whether the new model is truly utilizing the same knowledge as the original model, rather than just memorizing the training data.

Observations

- ToAlign + HDA outperforms all previous method and achieves state of art performance, i.e; it outperforms HDA by **0.9%.**

- Occupies almost the same GPU memory as the baseline i.e; it takes around **590 ms** and **6668 MB** GPU memory for one iteration.

- Baseline sometimes focuses on the background features which are useless to the image classification task, since it aligns the holistic features without considering the discriminativeness of different channels/sub-features.

- Our proposed ToAlign emphasizes that domain alignment task should assist/serve classification task, where we perform alignment under the guidance of the meta-knowledge induced from classification task.

References

1. Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In ICML, pages 1180–1189, 2015.
2. Z. Pei, Z. Cao, M. Long, and J. Wang. Multi-adversarial domain adaptation. In AAAI, volume 32, 2018.
3. S. Cui, X. Jin, S. Wang, Y. He, and Q. Huang. Heuristic domain adaptation. In NeurIPS, 2020.
4. G. Wei, C. Lan, W. Zeng, and Z. Chen. Metaalign: Coordinating domain alignment and classification for unsupervised domain adaptation. In CVPR, 2021.
5. Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky.
6. Domain-adversarial training of neural networks. Journal of Machine Learning Research, 17(1):2096–2030, 2016.

THANK YOU