BOSTON UNIVERSITY

Domain Adversarial Training

Anirudh Mandahr

CS664 Artificial Intelligence

Plan

- Introduction
- Datasets
- Deep Convolutional Neural Network
- Domain Adversarial Neural Network
- Multi-Adversarial Domain Adaptation
- Experiments
- Results
- Difficulties
- Conclusion
- References

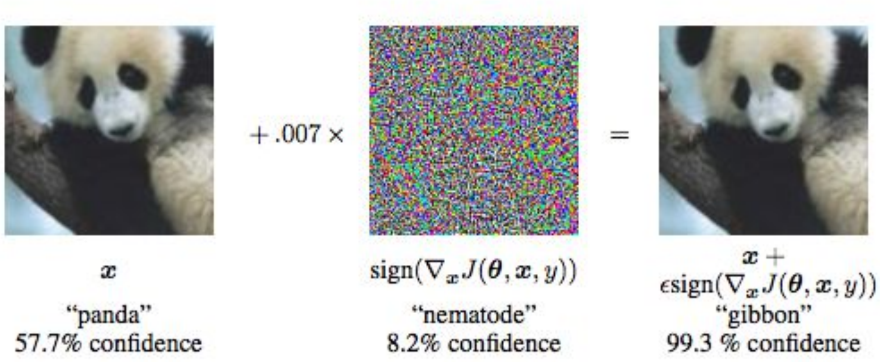
Introduction

Problem Statement:

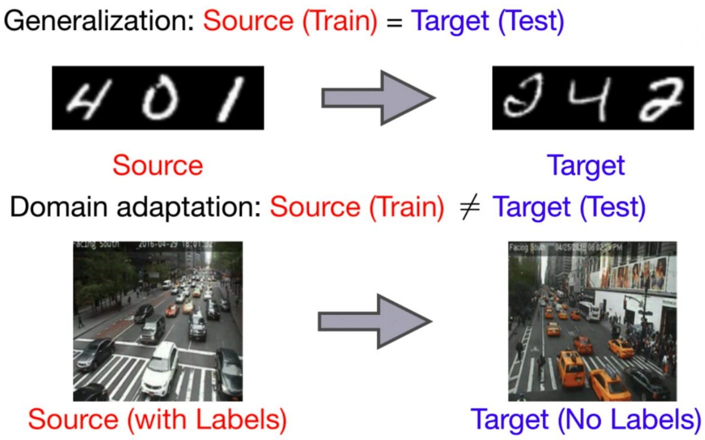
Deep convolutional neural network tends to be **weak at generalizing** on any type of image distribution and are therefore very depends on their training dataset distribution. This **lack of robustness** makes CNN easily fooled by adversarial examples.

The goal of **Adversarial Training** is to force a model to be less sensitivity to adversarial examples and therefore be **more robust**. One very interesting adversarial training is relative to domain distribution. In the real world, we often want to **adapt from a source domain labeled dataset** to different **target domain unlabeled datasets**.

Adversarial Example [1]



Domain Adaptation



Datasets

MNIST [2]

MNIST-M [3]

Both MNIST and MNIST-M

- 10 Classes
- (28x28x3)
- 50,000 Training examples
- 10,000 Validation examples
- 10,000 Test examples

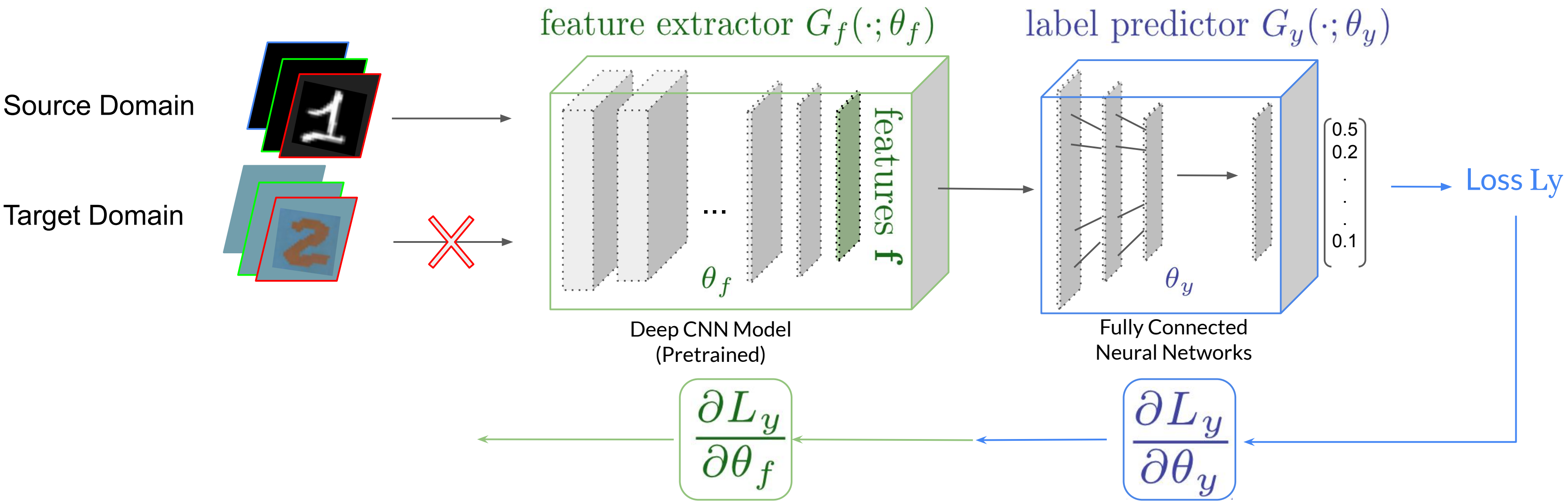
OFFICE-31 [4]

- 31 Classes
- Train 80%, Val 10%, Test 10%
- Amazon - 2,817 examples ($300 \times 300 \times 3$)
- DSLR - 498 examples($1000 \times 1000 \times 3$)
- Webcam - 795 examples (423×423x3)

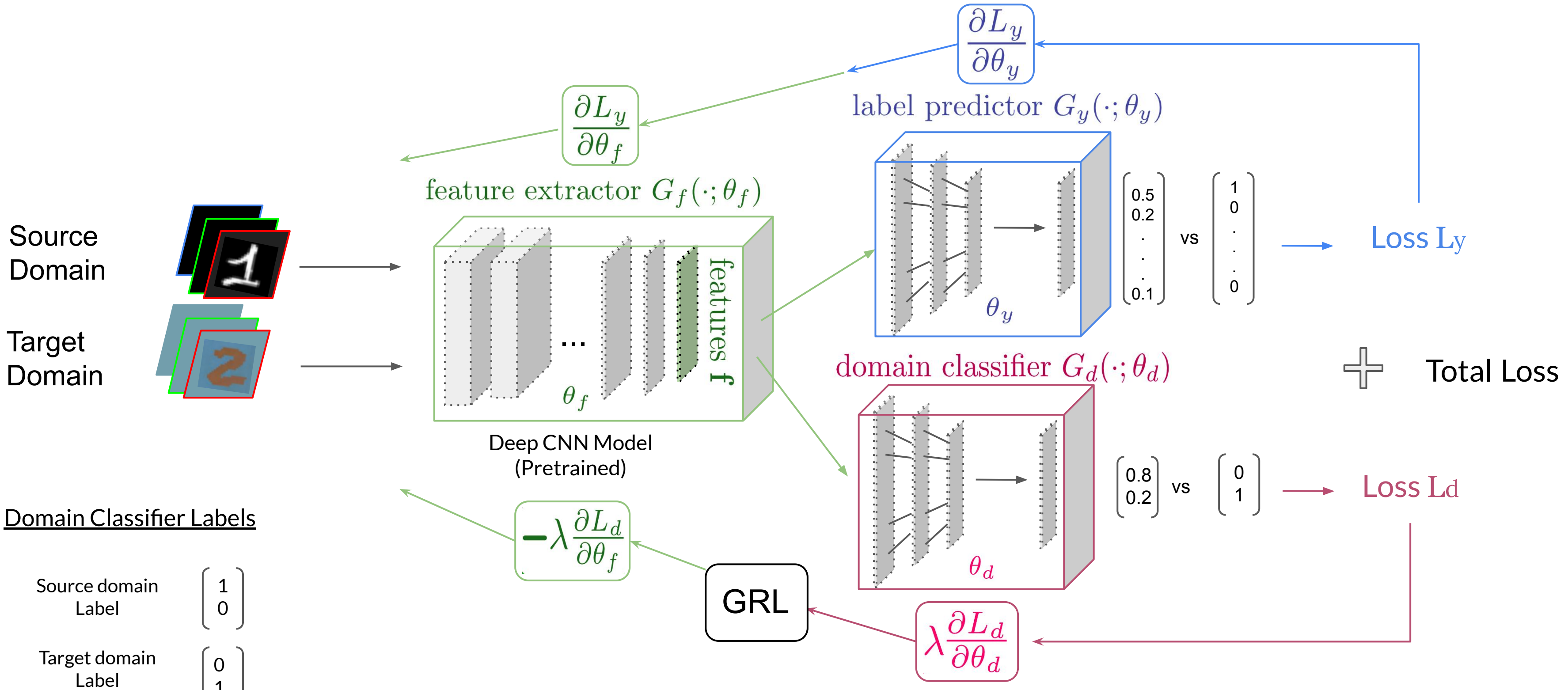
Deep Convolutional Neural Network

- Uses Transfer Learning [5]
- Only trained on Source dataset
- Evaluate at test time on Target dataset

- Feature f will not learn the target representation
- This DCNN will not perform well on any other domain distribution than its source
- Model is not robust to adversarial examples



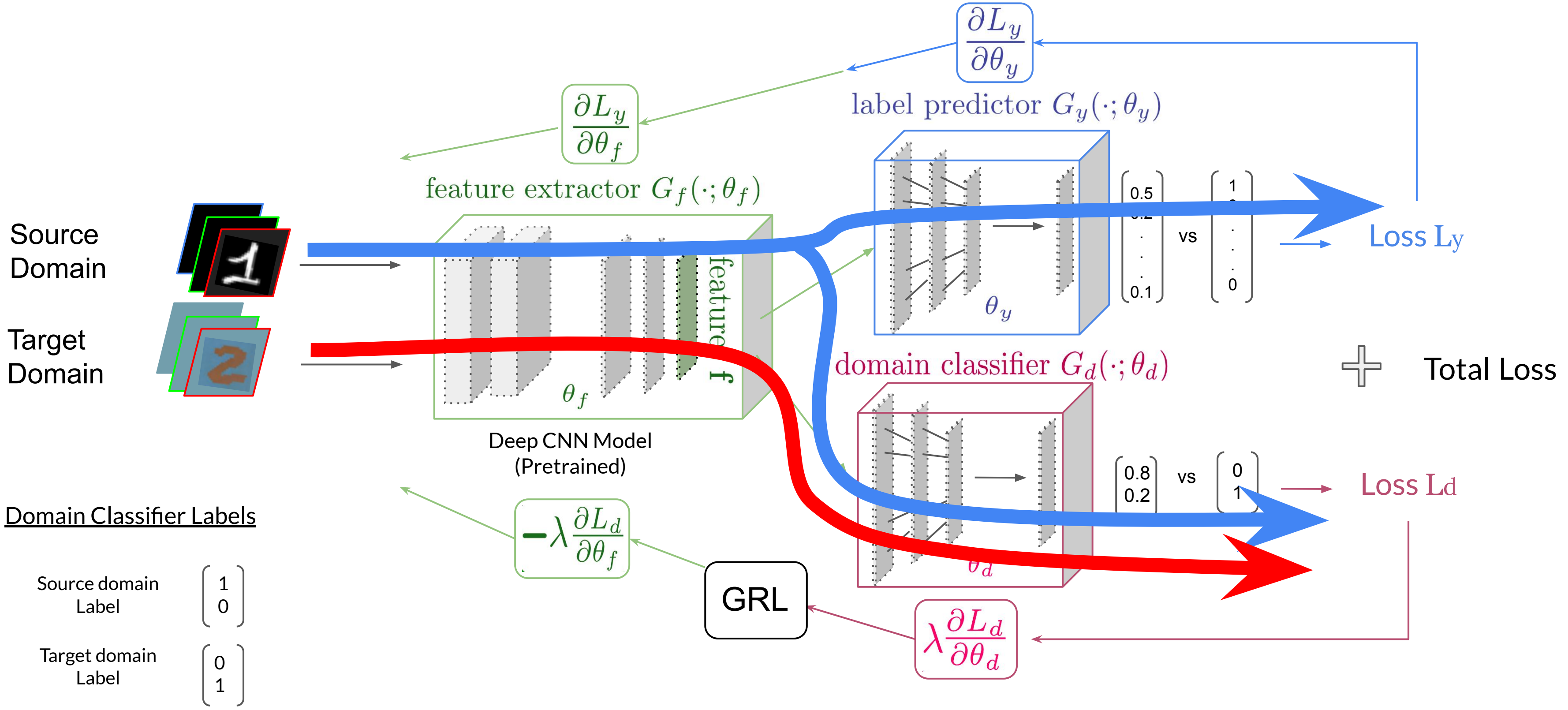
Domain Adaptation Neural Network [6]



<u>Domain Classifier Labels</u>

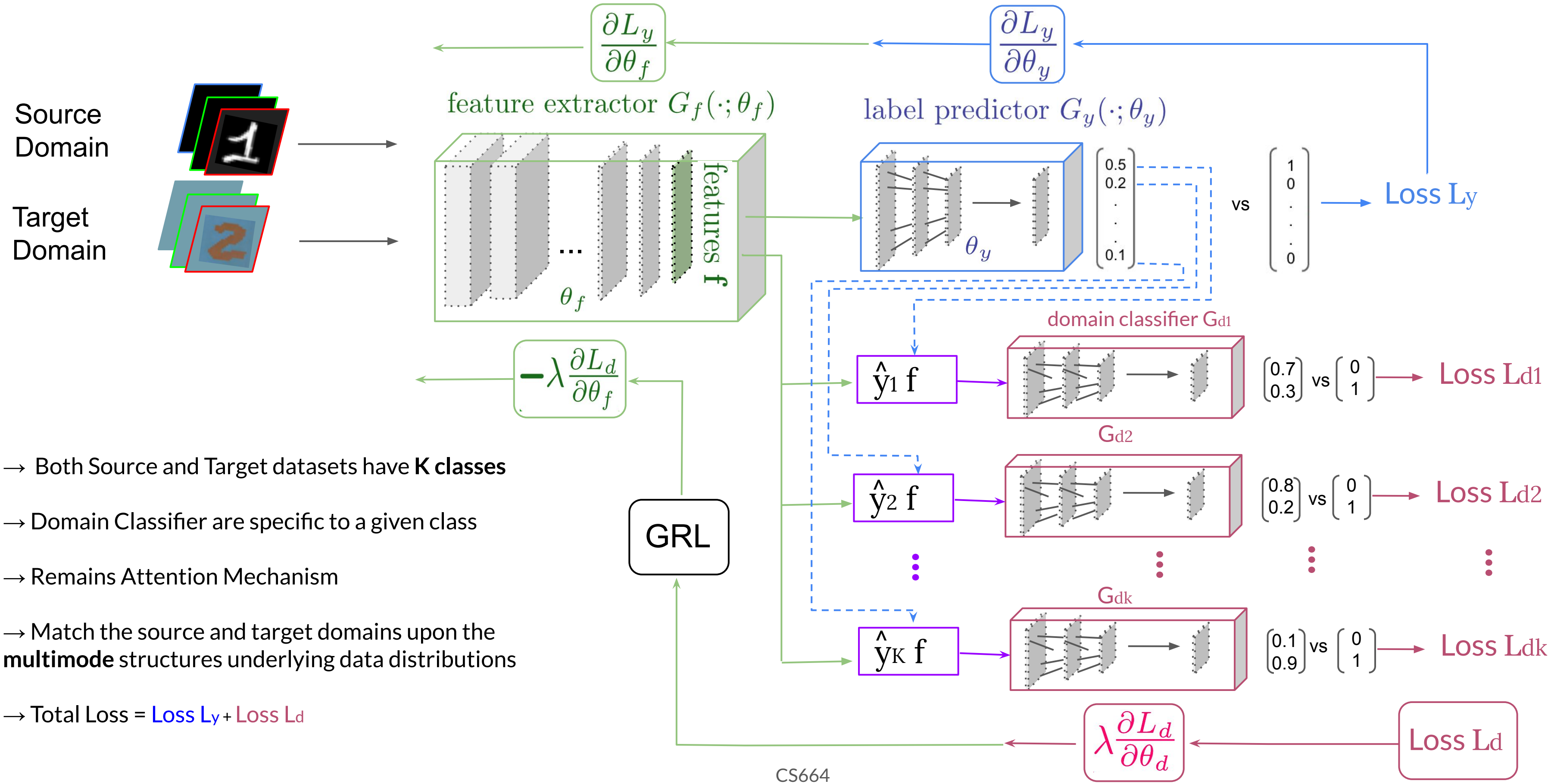
Source domain Label $\begin{pmatrix}1\\0\end{pmatrix}$

Target domain Label $\begin{pmatrix}0\\1\end{pmatrix}$



$\frac{\partial L_y}{\partial \theta_y}$
$\frac{\partial L_y}{\partial \theta_f}$
label predictor $G_y(\cdot;\theta_y)$
feature extractor $G_f(\cdot;\theta_f)$
Source Domain
Target Domain
feature f
θ_f
θ_y
0.5
0.2
.
.
.
0.1
vs
1
0
.
.
.
0
Loss Ly
domain classifier $G_d(\cdot;\theta_d)$
Total Loss
Deep CNN Model (Pretrained)
0.8
0.2
vs
0
1
Loss Ld
θ_d
Domain Classifier Labels
Source domain Label
1
0
Target domain Label
0
1
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
GRL
$\lambda\frac{\partial L_d}{\partial \theta_d}$

Multi-Adversarial Domain Adaptation [7]



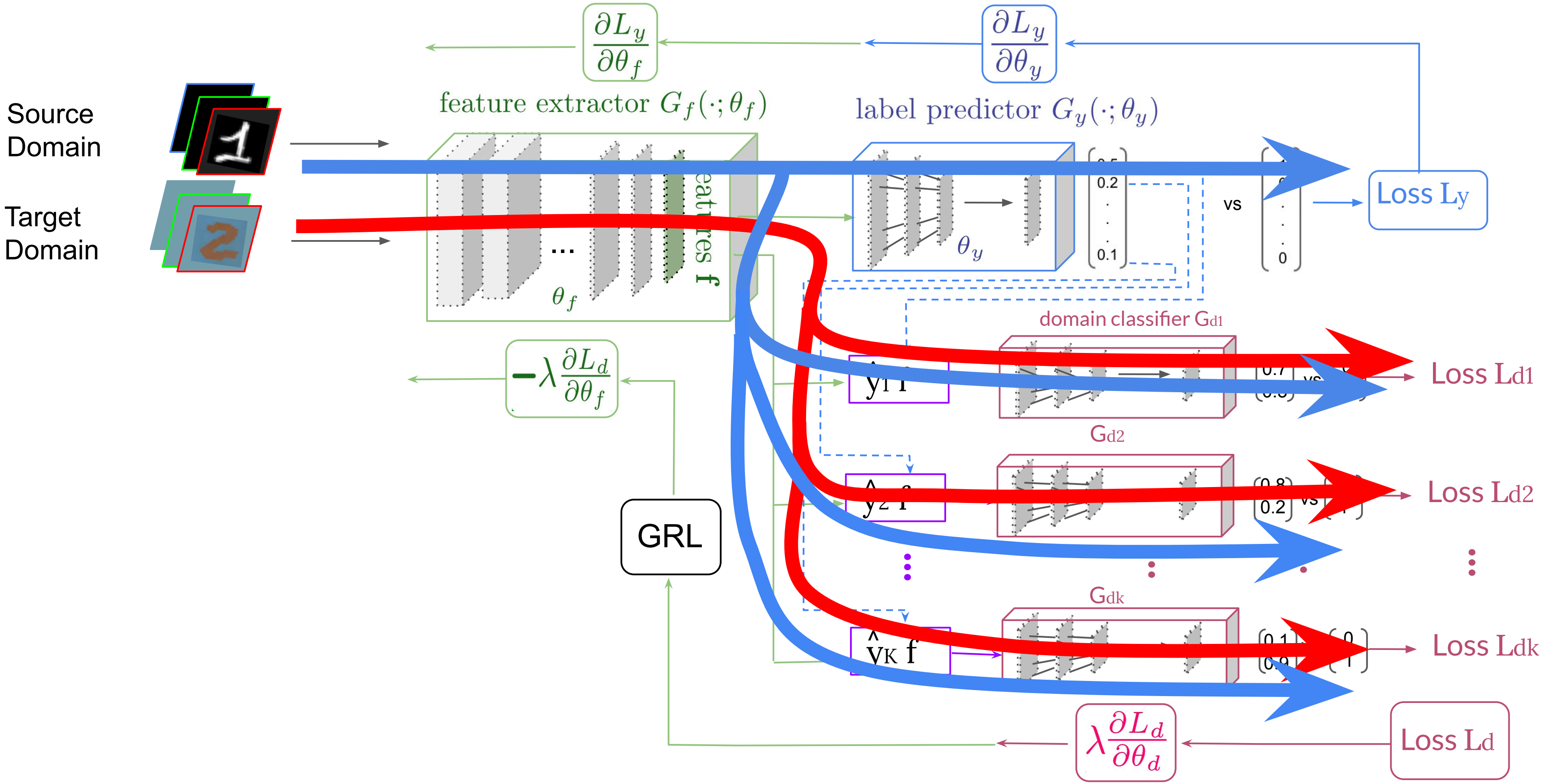
→ Both Source and Target datasets have **K classes**

→ Domain Classifier are specific to a given class

→ Remains Attention Mechanism

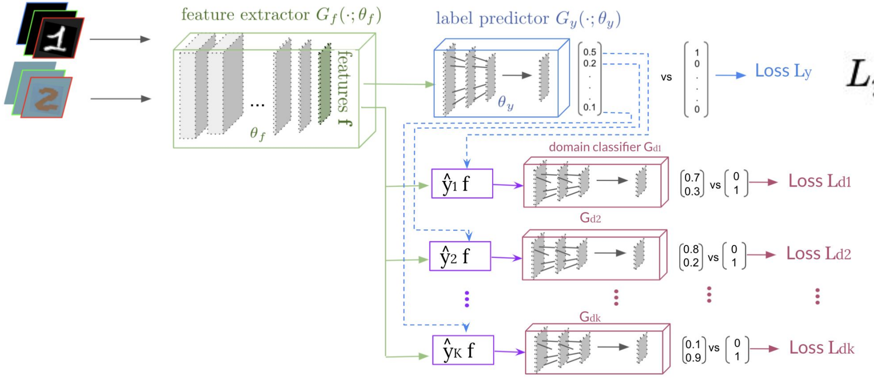
→ Match the source and target domains upon the **multimode** structures underlying data distributions

→ Total Loss = Loss L_y + Loss L_d


Source Domain
Target Domain
$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
feature extractor $G_f(\cdot;\theta_f)$
label predictor $G_y(\cdot;\theta_y)$
θ_f
θ_y
0.2
0.1
vs
0
Loss Ly
domain classifier Gd1
Loss Ld1
Gd2
Loss Ld2
Gdk
Loss Ldk
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
GRL
0.2
$\lambda\frac{\partial L_d}{\partial \theta_d}$
Loss Ld

$$\mathcal{D}_s = \{(\mathbf{x}_i^s, \mathbf{y}_i^s)\}_{i=1}^{n_s}$$

$$\mathcal{D}_t = \{\mathbf{x}_j^t\}_{j=1}^{n_t}$$



$$L_y\left(G_y\left(G_f\left(\mathbf{x}_i\right)\right), y_i\right)$$

$$\sum_{k=1}^{K} L_d^k\left(G_d^k\left(\hat{y}_i^k G_f\left(\mathbf{x}_i\right)\right), d_i\right)$$

$$C\left(\theta_f, \theta_y, \theta_d^k|_{k=1}^K\right) = \frac{1}{n_s} \boxed{\sum_{\mathbf{x}_i \in \mathcal{D}_s} L_y\left(G_y\left(G_f\left(\mathbf{x}_i\right)\right), y_i\right)} - \frac{\lambda}{n} \boxed{\sum_{k=1}^{K} \sum_{\mathbf{x}_i \in \mathcal{D}} L_d^k\left(G_d^k\left(\hat{y}_i^k G_f\left(\mathbf{x}_i\right)\right), d_i\right)}$$

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg \min_{\theta_f, \theta_y} C\left(\theta_f, \theta_y, \theta_d^k|_{k=1}^K\right),$$

$$(\hat{\theta}_d^1, \ldots, \hat{\theta}_d^K) = \arg \max_{\theta_d^1, \ldots, \theta_d^K} C\left(\theta_f, \theta_y, \theta_d^k|_{k=1}^K\right)$$

Experiments

PyTorch Lightning [8]

PyTorch

Weights & Biases [9]

docker

NVIDIA

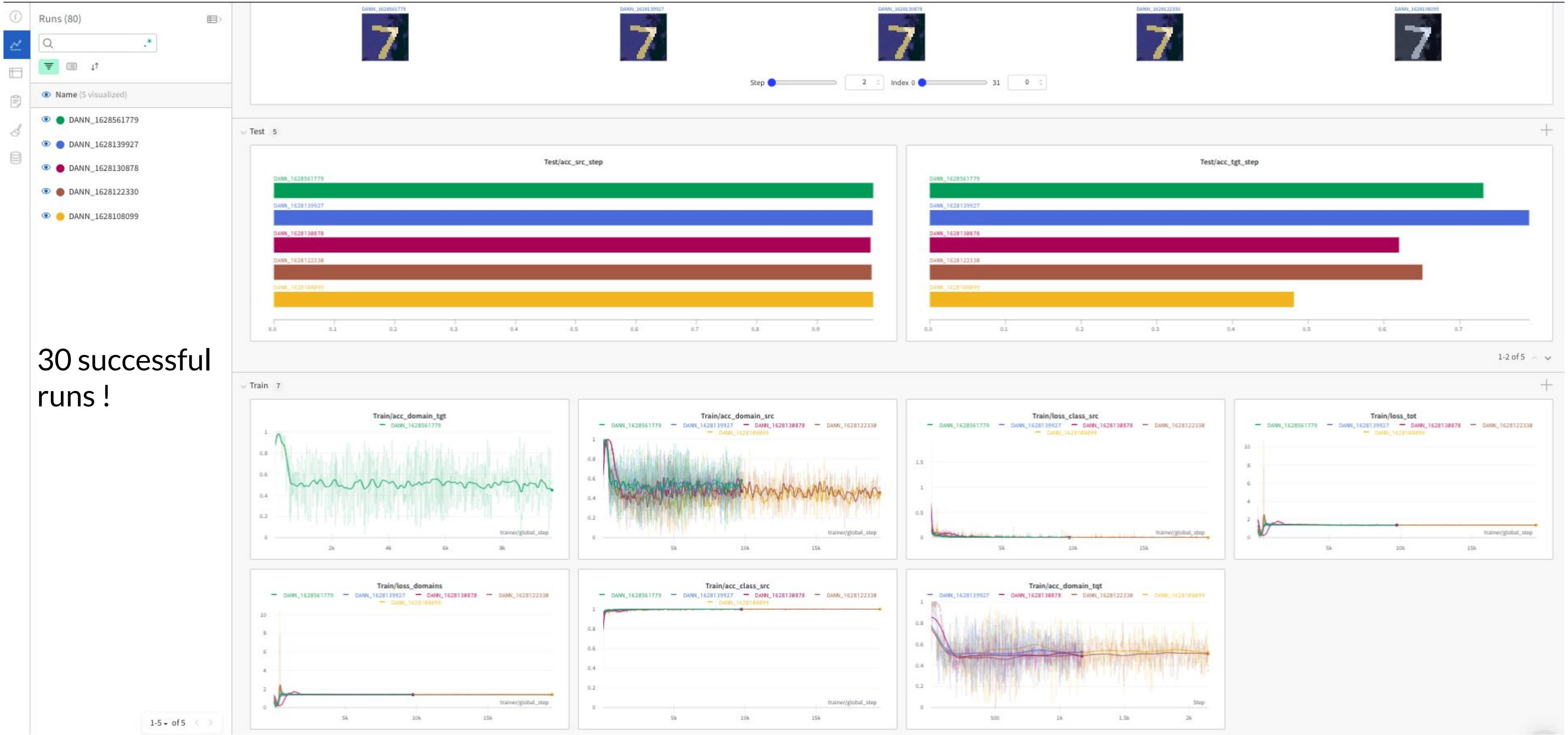
Experiment configuration files

config.yml

```yaml
input:
  dataset:
    src: "AMAZON" # Source Dataset name(MNIST or WEBCAM)
    tgts: ["DSLR"] # Target Datasets name, please do not provide several targets for MADA.
    transformation :
        img_size: 224 # size of input if images input
        src : transform_RGB_DA # name of the transform to perform on source data
        tgt : transform_RGB_DA # name of the transform to perform on target data
        mean: [0.485, 0.456, 0.406] # mean used for normalization (from resnet)
        std: [0.229, 0.224, 0.225] # std used for normalization (from resnet)

model:
  type : DANN # MADA, DANN
  backbone: resnet34 #resnet18, resnet34, resnet152
  pretrained_backbone: imagenet # if not imagenet then not pretrained
  n_layers_freeze: 0 # Depends on your backbone
  class_classifier: linear3_bn2_v1 # linear2_dr2_bn, linear2_bn, linear3_bn2_v1, linear3_bn2_v2
  domain_classifier: linear3_bn2_v2

training:
  gpus: 1
  num_workers: 0
  optimizer:
    type: Adam #Adam, SGD
    momentum: 0.9
    lr: 0.001
    weight_decay: 2.5e-5
  scheduler:
    lr_schedule: true
    alpha: 10
    gamma: 10
    beta: 0.75
  batch_size: 256
  epochs: 50

seed: 8888 #random seed for reproducibilty
```

Experiments Tracking and Iterations

Weights & Biases



30 successful runs !

Experiment Run Example

Results

Classification accuracies between source and target domain for MNIST and OFFICE 31 Datasets

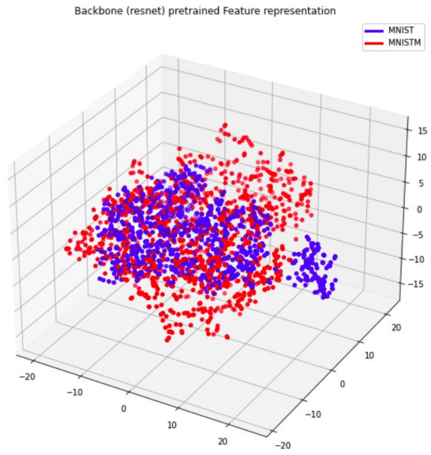
	MNIST → MNISTM		AMAZON → DSLR		AMAZON → WEBCAM	
	Test Source Accuracy	Test Target Accuracy	Test Source Accuracy	Test Target Accuracy	Test Source Accuracy	Test Target Accuracy
Our DCNN (Train on Source only)	.9916	.265 (.5225)	.660	**.600** (.689)	.762	**.600** (.684)
Our DANN (from Ganin, Lempitsky et al 2015)	.9918	.7917 (.7666)	.760	**.600** (.797)	.700	**.600** (.820)
Our MADA (from Pei, Zhongyi, et al 2018)	.871	.3705	.740	**.630** (.878)	.680	**.650** (.90)
Our DCNN (Train on Target only)	.973 (.959)	.957	.360	.159	.987	.587

(*) from https://arxiv.org/pdf/1809.02176.pdf (MADA), https://arxiv.org/pdf/1505.07818v4.pdf (DANN).

*Feature extractor is ResNet18 /ResNet34/ResNet 152 fully retrained.
*Optimizer is SGD or Adam, Learning rate 0.001.
*FCNN are 2/3 dense layers followed by BatchNorm / Dropout.

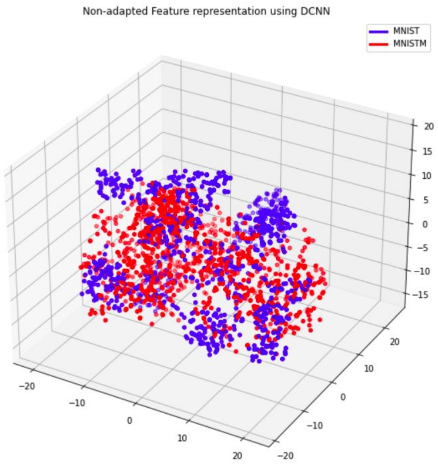
Top feature extractor visualisation using TSNE (MNIST and MNISTM Test Dataset)

RESNET18



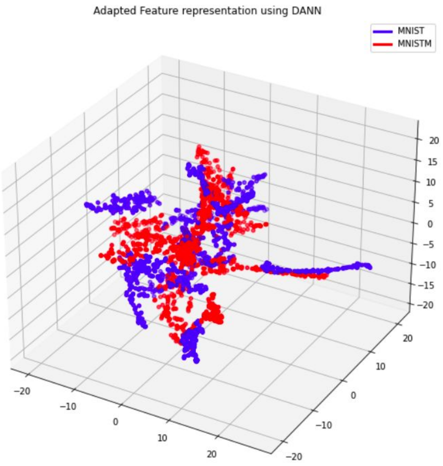
- Source: No cluster per class
- Target: No cluster per class
- Distribution separated

DCNN



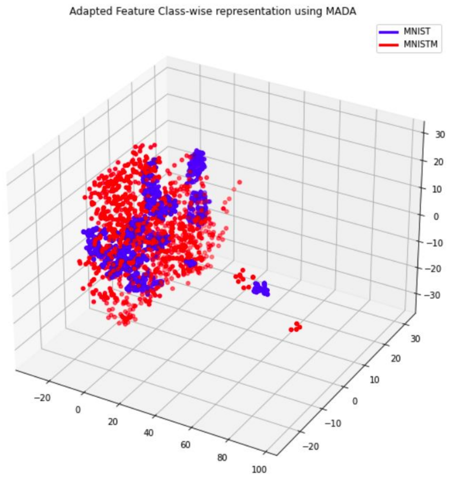
- Source: **clusters** per class
- Target: No cluster per class
- Distribution separated

DANN



- Source: **clusters** per class
- Target: **clusters** per class
- Distribution **Mixed**

MADA



- Source: **clusters** per class
- Target: No clusters per class
- Distribution mostly separated

Difficulties

- Unbalanced dataset repartition between Source and Target.
- Unstable Training between domain classifier and feature extractor.
- Requires more GPU Memory to load both Source and Target simultaneously.
- Optimizing using a single Optimizer (unlike GAN's)
- MADA has only very few implementation on the web (only in C++ from the original paper)

Conclusion

What we have learned:

- Good practices for ML Project Experimentation
- Better read Deep Learning papers
- Implement only by looking the paper
- Research is difficult!

Future work:

- Try different batch loading techniques for MADA to fix the unbalanced issues
- Try to use 2 separate Optimizers for MADA
- Use different Pretrained Feature Extractors (VGG, Xception, ViT, …)
- Try MDANN (domain classifiers are learned for K multiple Domains)

References

[1] Stanford CS231 Lecture 16 | Adversarial Examples and Adversarial Training (Ian Goodfellow)
https://www.youtube.com/watch?v=CIfsB_EYsVI&t=4367s

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, November 1998.

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. IEEE Transaction Pattern Analysis and Machine Intelligence, 33, 2011.

[4] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In ECCV, pages 213–226, 2010.

[5] Tan, Chuanqi, et al. "A survey on deep transfer learning." *International conference on artificial neural networks*. Springer, Cham, 2018. https://arxiv.org/abs/1808.01974

[6] Ganin, Yaroslav, et al. "Domain-adversarial training of neural networks." *The journal of machine learning research* 17.1 (2016): 2096-2030. https://arxiv.org/abs/1505.07818

[7] Pei, Zhongyi, et al. "Multi-adversarial domain adaptation." *Thirty-second AAAI conference on artificial intelligence*. 2018. https://arxiv.org/abs/1809.02176

[8] Falcon, W. "PyTorchLightning/pytorch-lightning." *Pytorch Lightning* (2021).

[9] L. Biewald, "Experiment Tracking with Weights and Biases," Weights & Biases. [Online]. Available: http://wandb.com/. [Accessed: 07/2021].