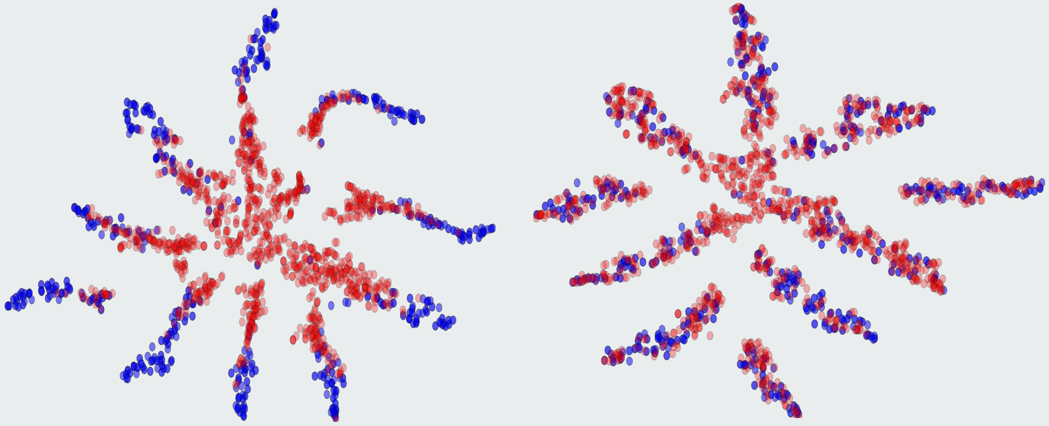
Unsupervised Domain Adaptation by Backpropagation

Chih-Hui Ho, Xingyu Gu, Yuan Qi



Outline

- Introduction
- Architecture
  - Baseline architecture
  - GAN architecture
  - WGAN architecture
- Experiments
- Conclusions

Introduction

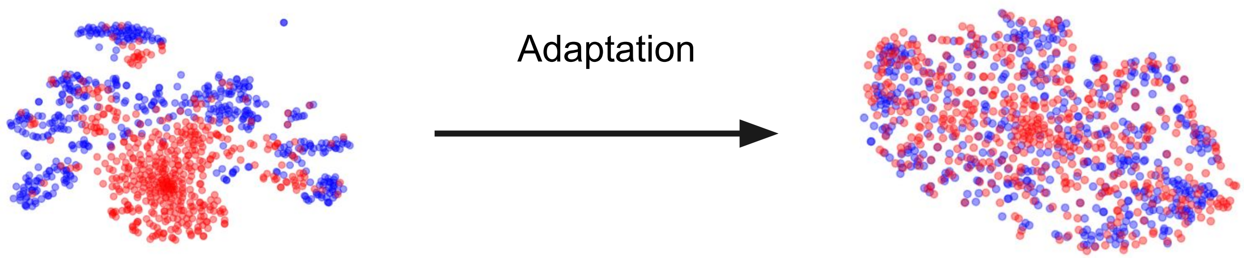
Deep network: requires massive **labeled** training data.

- **Difficult to collect** sometimes:
    - Robotics
    - Disaster
    - Medical diagnosis
    - Bioinformatics

Domain Adaptation solves this problem by using relevant datasets: i.e. training on the data from source domain, but testing on the data from target domain.

Example

- MNIST → MNIST-M (extracted features)



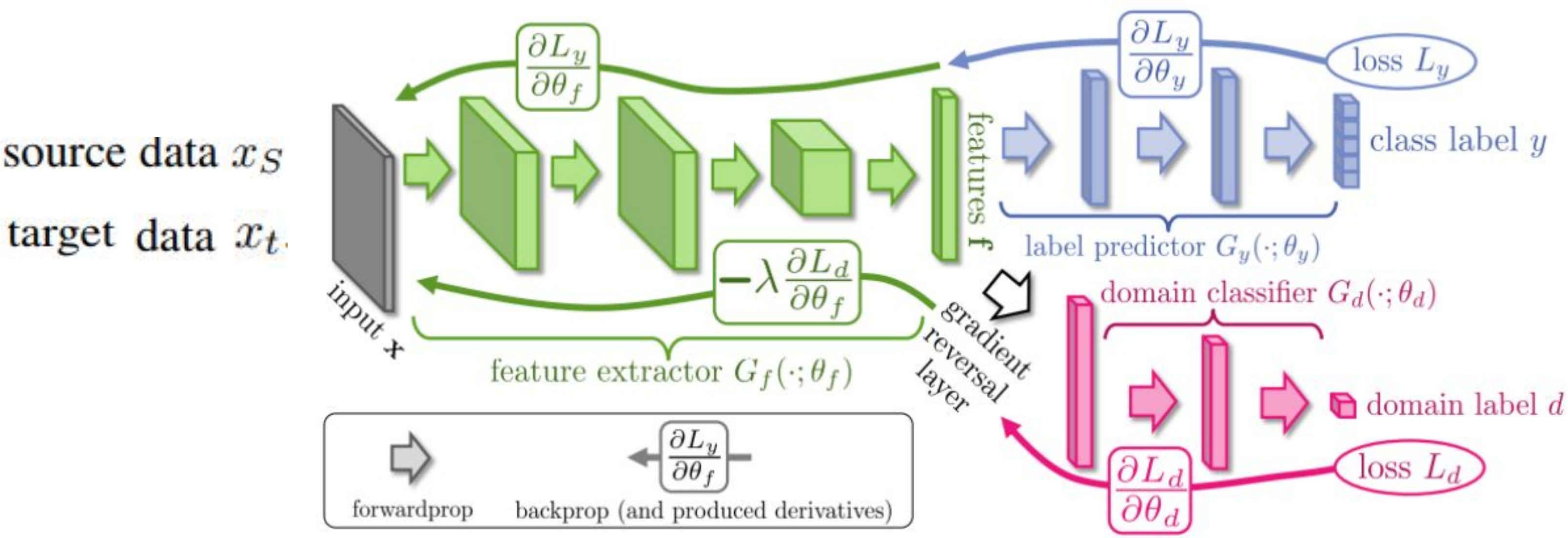
Our Implementation

- **Three architectures:**
  - Baseline architecture (Y. Ganin et al., 2015)
  - GAN architecture
  - WGAN architecture

- **Five experiments:**
  - MNIST → MNIST-M
  - SYN NUM → SVHN
  - SVHN → MNIST
  - SYN SIGN → GTSRB
  - Office Dataset
    - Amazon → Webcam
    - DSLR → Webcam
    - Webcam → DSLR

Baseline architecture

- G_f : feature extractor
- G_y : label predictor
- G_d : domain classifier

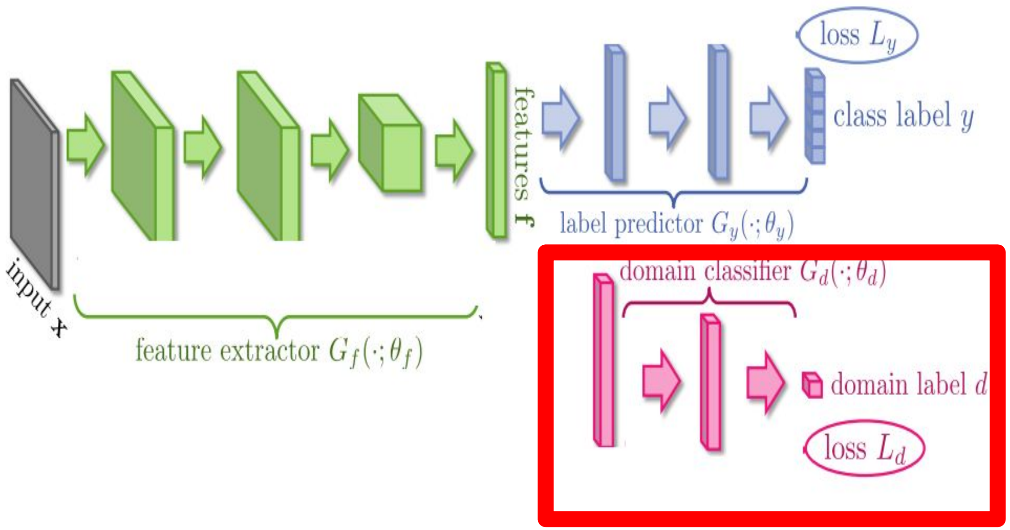


Source → 0
Target → 1

GAN based architecture

- Incorporate the generative adversarial network into loss function

$$\underset{\theta_d}{\operatorname{argmax}}\ \underset{\theta_f,\theta_y}{\operatorname{argmin}}\ log(G_d(G_f(x_t))) + log(1 - G_d(G_f(x_s))) + G_y(G_f(x_s), y_s)$$



Train

Algorithm 1 GAN based domain adaptation algorithm

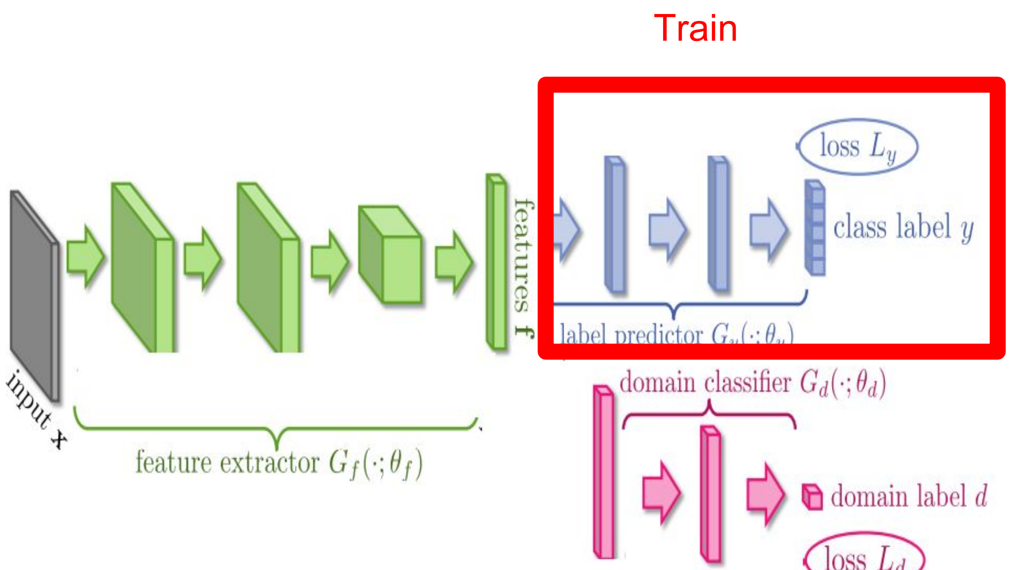
Randomly initialize the network
Define $L1 = log(G_d(G_f(x_t)))$
Define $L2 = log(1 - G_d(G_f(x_s)))$
Define $L3 = L(G_f(x_s), y_s)$

for e < MaxEpoch **do**
  `Fix` θ_f `and` θ_y
  $\theta_d \leftarrow \theta_d - \mu\frac{\partial(-L1-L2)}{\partial\theta_d}$
  `Fix` θ_f `and` θ_d
  $\theta_y \leftarrow \theta_y - \mu\frac{\partial L3}{\partial\theta_y}$
  `Fix` θ_y `and` θ_d
  $\theta_f \leftarrow \theta_f - \mu\frac{\partial(L1+L2+\lambda L3)}{\partial\theta_f}$
end for

GAN based architecture

- Incorporate the generative adversarial network into loss function

$$\underset{\theta_d}{\text{argmax}}\ \underset{\theta_f,\theta_y}{\text{argmin}}\ log(G_d(G_f(x_t))) + log(1 - G_d(G_f(x_s))) + G_y(G_f(x_s), y_s)$$



Algorithm 1 GAN based domain adaptation algorithm

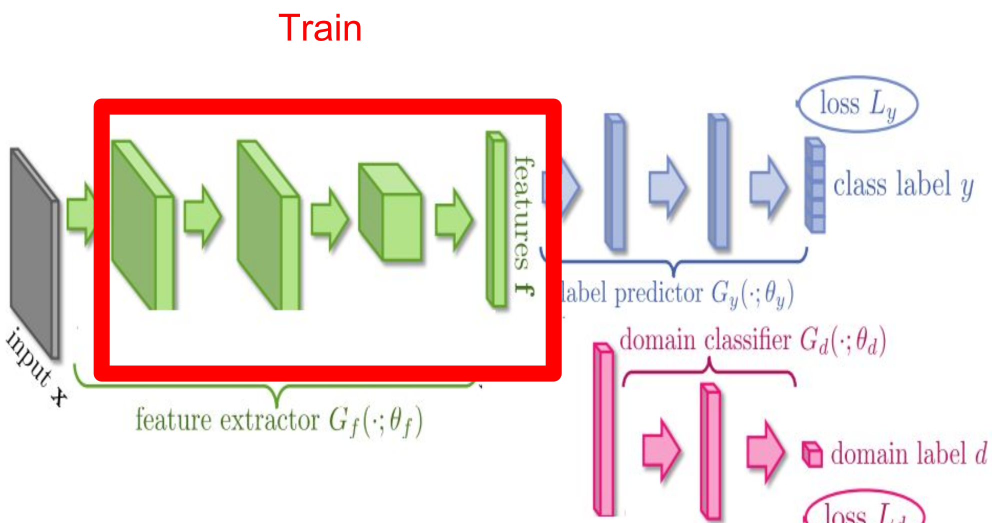
Randomly initialize the network
Define $L1 = log(G_d(G_f(x_t)))$
Define $L2 = log(1 - G_d(G_f(x_s)))$
Define $L3 = L(G_f(x_s), y_s)$

for e < MaxEpoch **do**
  Fix θ_f and θ_y
  $\theta_d \leftarrow \theta_d - \mu\frac{\partial(-L1-L2)}{\partial\theta_d}$
  Fix θ_f and θ_d
  $\theta_y \leftarrow \theta_y - \mu\frac{\partial L3}{\partial\theta_y}$
  Fix θ_y and θ_d
  $\theta_f \leftarrow \theta_f - \mu\frac{\partial(L1+L2+\lambda L3)}{\partial\theta_f}$
end for

GAN based architecture

- Incorporate the generative adversarial network into loss function

$$\underset{\theta_d}{\operatorname{argmax}}\ \underset{\theta_f,\theta_y}{\operatorname{argmin}}\ log(G_d(G_f(x_t))) + log(1 - G_d(G_f(x_s))) + G_y(G_f(x_s), y_s)$$



Algorithm 1 GAN based domain adaptation algorithm

Randomly initialize the network

Define $L1 = log(G_d(G_f(x_t)))$

Define $L2 = log(1 - G_d(G_f(x_s)))$

Define $L3 = L(G_f(x_s), y_s)$

for e < MaxEpoch **do**

  Fix θ_f and θ_y

  $\theta_d \leftarrow \theta_d - \mu\frac{\partial(-L1-L2)}{\partial\theta_d}$

  Fix θ_f and θ_d

  $\theta_y \leftarrow \theta_y - \mu\frac{\partial L3}{\partial\theta_y}$

  Fix θ_y and θ_d

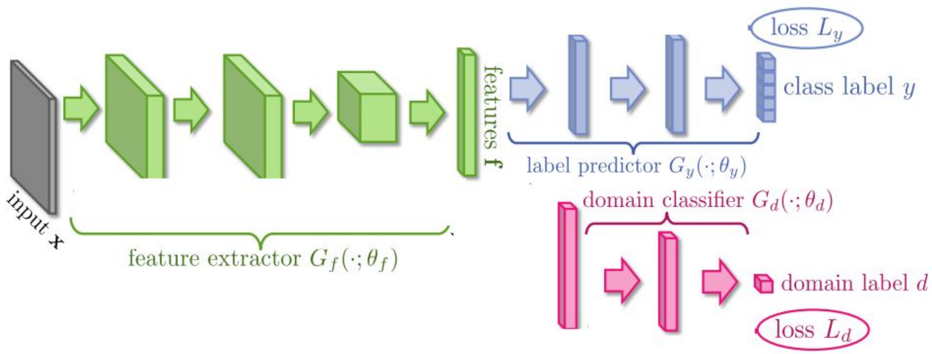
  $\theta_f \leftarrow \theta_f - \mu\frac{\partial(L1+L2+\lambda L3)}{\partial\theta_f}$

end for

WGAN architecture

- Slightly modified the loss function

$$\underset{\theta_d}{\operatorname{argmax}}\ \underset{\theta_f,\theta_y}{\operatorname{argmin}}\ G_d(G_f(x_t)) - G_d(G_f(x_s)) + G_y(G_f(x_s), y_s)$$



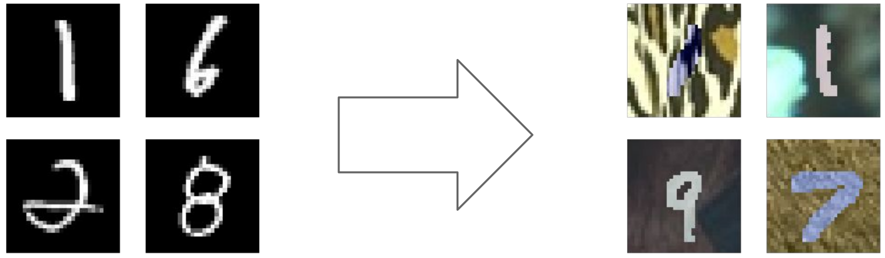
Algorithm 2 WGAN based domain adaptation algorithm

Randomly initialize the network
Define $L1 = G_d(G_f(x_t))$
Define $L2 = -G_d(G_f(x_s))$
Define $L3 = L(G_f(x_s), y_s)$

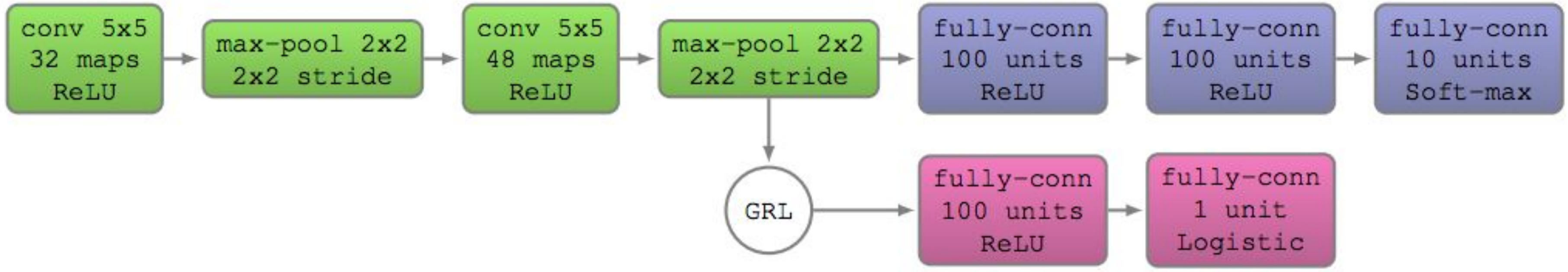
for e < MaxEpoch **do**
  `Fix` θ_f `and` θ_y
  $\theta_d \leftarrow \theta_d - \mu\frac{\partial(-L1-L2)}{\partial\theta_d}$
  `Fix` θ_f `and` θ_d
  $\theta_y \leftarrow \theta_y - \mu\frac{\partial L3}{\partial\theta_y}$
  `Fix` θ_y `and` θ_d
  $\theta_f \leftarrow \theta_f - \mu\frac{\partial(L1+L2+\lambda L3)}{\partial\theta_f}$
end for

Experiments - MNIST → MNIST-M

- Example



- Network Architecture

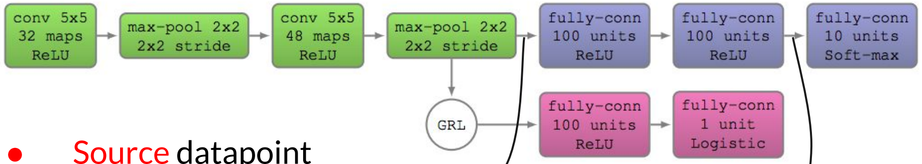


Experiments - MNIST → MNIST-M

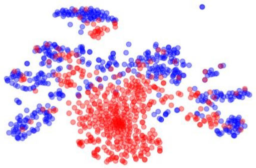
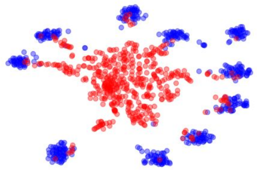
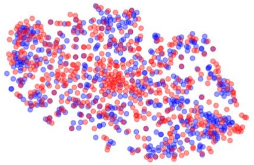
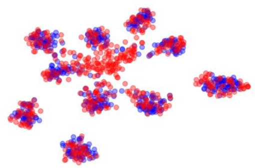
- Result

Method	Ours	Original paper
Source only	0.551	0.523
Proposed approach	0.872	0.767
Train on target	0.963	0.960

- Visualization (t-SNE)

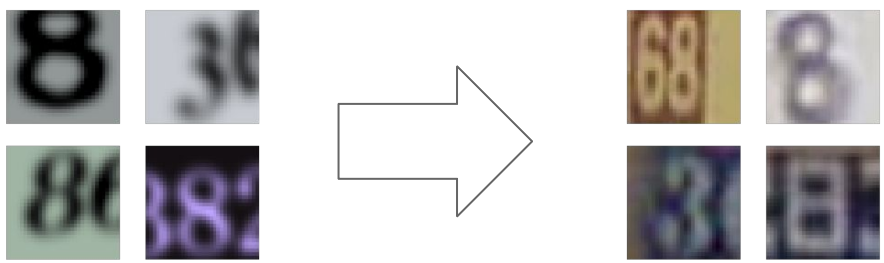


- Source datapoint
- Target datapoint

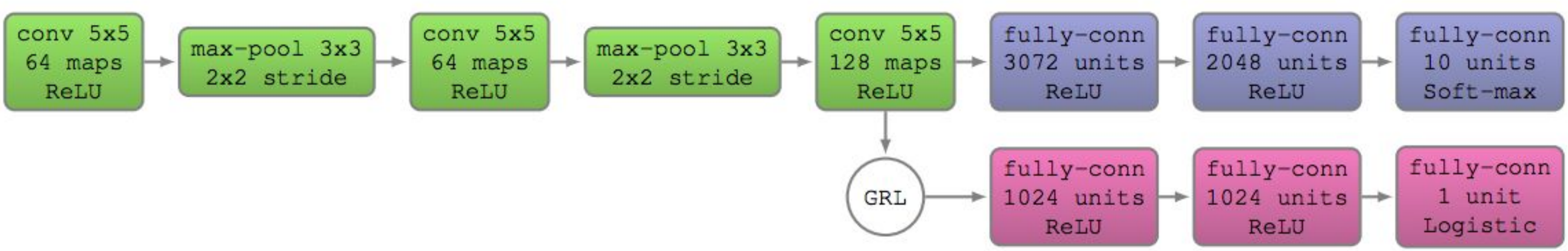
Method	Feature	Last hidden layer
Source only		
Proposed approach		

Experiments - SYN NUM → SVHN

- Example



- Network Architecture

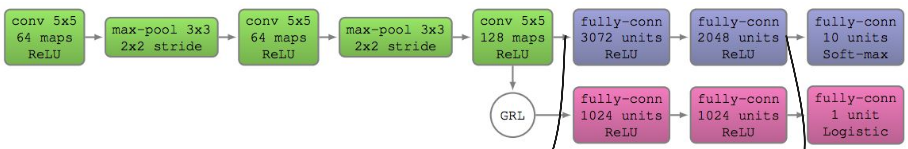


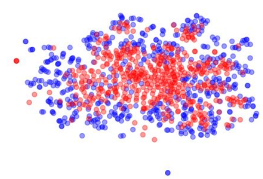
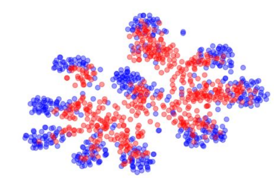
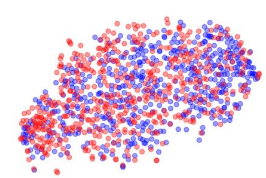
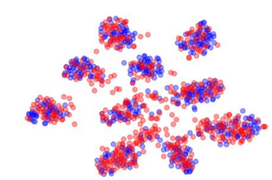
Experiments - SYN NUM → SVHN

- Result

Method	Ours	Original paper
Source only	0.880	0.867
Proposed approach	0.894	0.911
Train on target	0.927	0.922

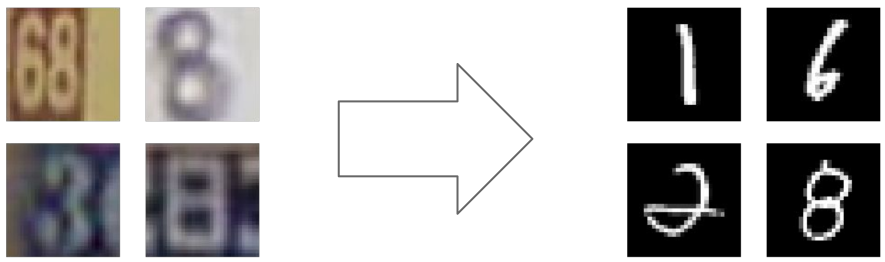
- Visualization



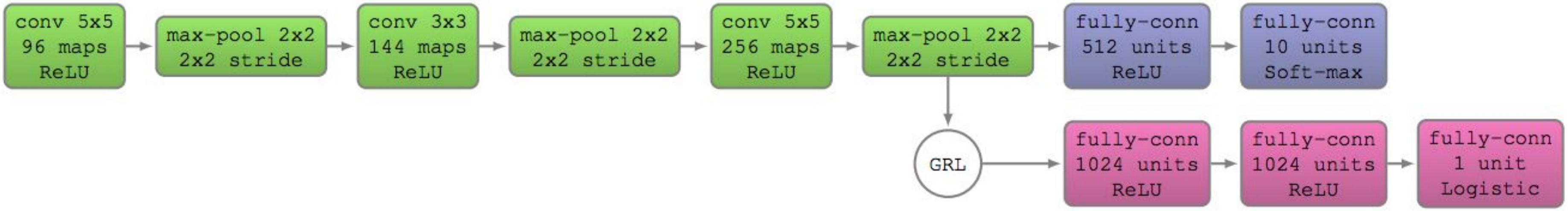
Method	Feature	Last hidden layer
Source only		
Proposed approach		

Experiments - SVHN → MNIST

- Example



- Network Architecture



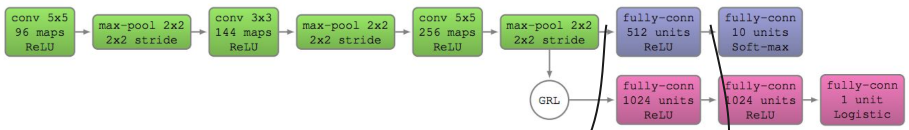
Experiments - SVHN → MNIST

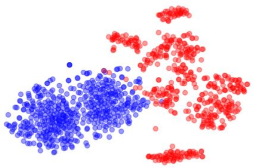
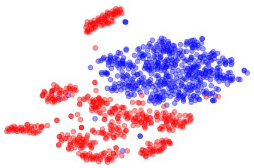
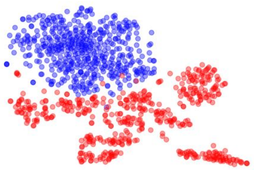
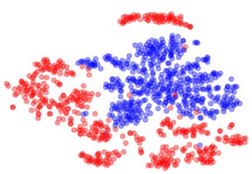
- Result

Method	Ours	Original paper
Source only	0.604	0.549
Proposed approach	0.698	0.739
Train on target	0.992	0.994

- Problem:
  Label classifier too weak
  Feature extractor too weak

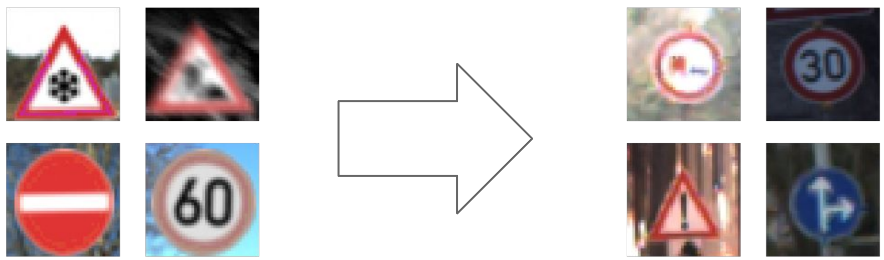
- Visualization



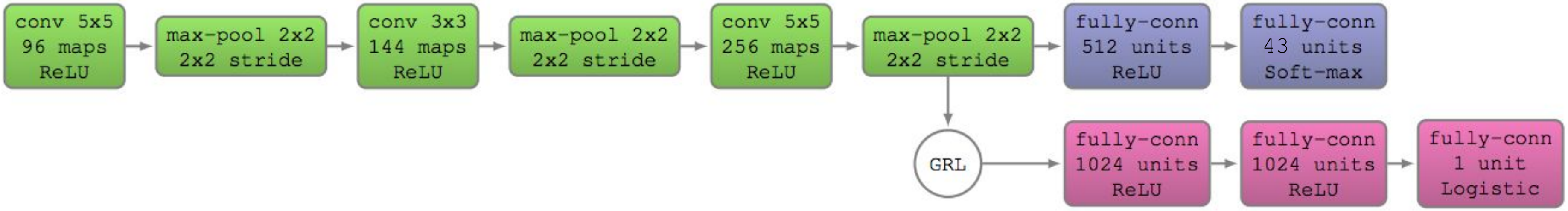
Method	Feature	Last hidden layer
Source only		
Proposed approach		

Experiments - SYN SIGN → GTSRB (ongoing)

- Example



- Network Architecture



Experiments - SYN SIGN → GTSRB (ongoing)

- Current Result

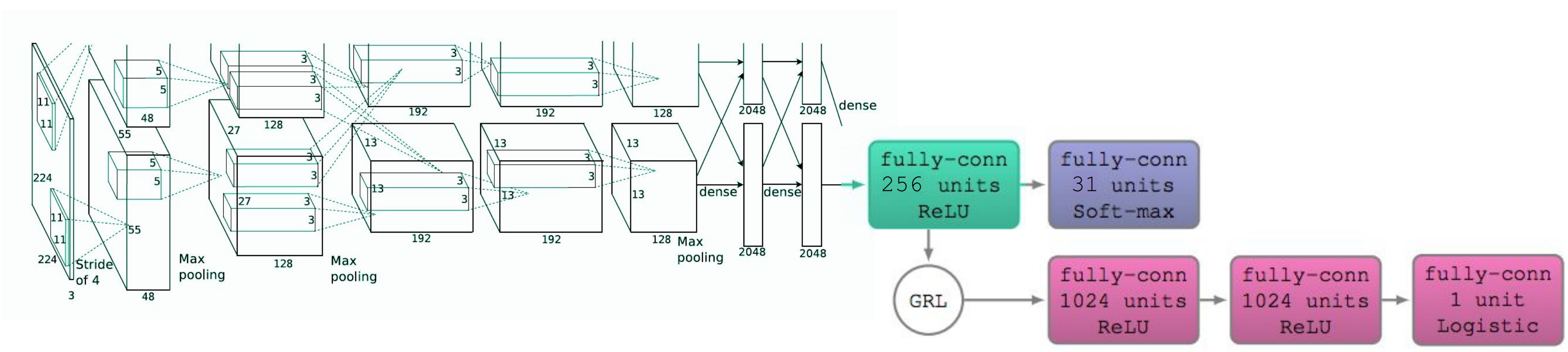
Method	Ours	Original paper
Source only	0.563	0.790
Proposed approach	---	0.887
Train on target	0.555	0.998

Experiments - Office Dataset (Amazon, Webcam, DSLR)

- Example

Amazon Webcam DSLR

- Architecture (pretrianed AlexNet + Label classifier + Domain classifier)



Experiments - Office Dataset

- Result

Method (Source / Target)	Amazon → Webcam	DSLR → Webcam	Webcam → DSLR
Source only | (Original paper)	0.506 | (0.642)	0.914 | (0.961)	0.967 | (0.978)
Proposed approach | (Original paper)	0.565 | (0.730)	0.934 | (0.964)	0.989 | (0.992)

- Improvement in percentile: similar
- Start point (source only): ours are lower than the original paper's
  - Possible reason: different initialization, training process, etc.

Experiments - GAN & WGAN

- Problem
  - Hard to converge
- Current Result
  - DA accuracy similar to source only baseline (no improvement)
- Solution
  - Examine the implementation
  - Optimize without momentum (WGAN)

Conclusion

- First group to implement all the experiments in the original paper
- The datasets used in the paper are organized to be used with ease
- Investigate the possibility to incorporate GAN loss function
- All codes are released on github for future research