AI1603 Assignment Report

Recurrence and Exploration of OoD Algorithms

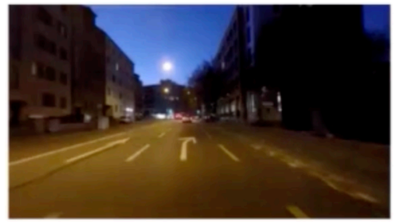
Bangjun Wong 2022.7.14

Cooperator: Xiangyuan Xue

Problem Setting

Backgrounds

- Neural Networks are often designed based on the hypothesis that all data are *Independent-Identically distributed*.
- In realistic situations, however, *Independent-Identically distributed* hypothesis is rarely true.
- Promising AI systems ought to still hold robust generalizing abilities under *Out-of-Distribution* circumstances.
- The key is to learn the casual features of the source dataset.



Twilight time-real

Nighttime-real

Twilight time-synthetic

Nighttime-synthetic

Problem Setting

Colored MNIST

Causal Feature

Environment	Size	Label Flipped	Color Flipped	Theoretical Accuracy
train1	20000	25%	20%	75%
train2	20000	25%	10%	75%
test	20000	25%	90%	75%

Table 1: Properties of ColoredMNIST

Spurious Feature

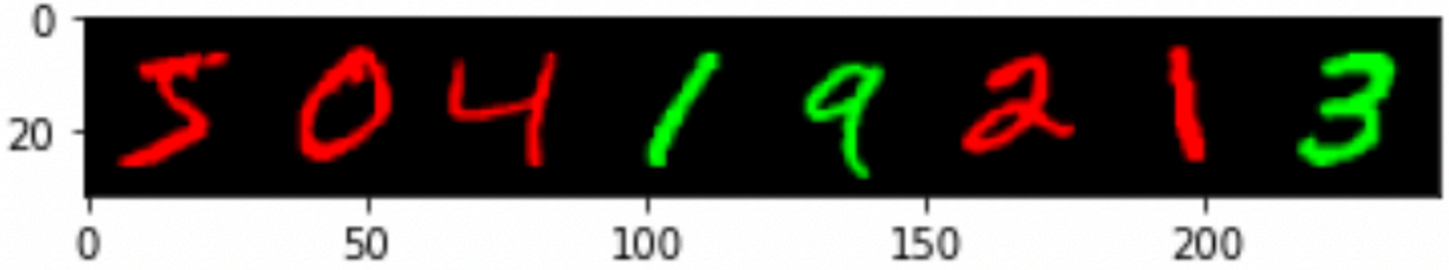


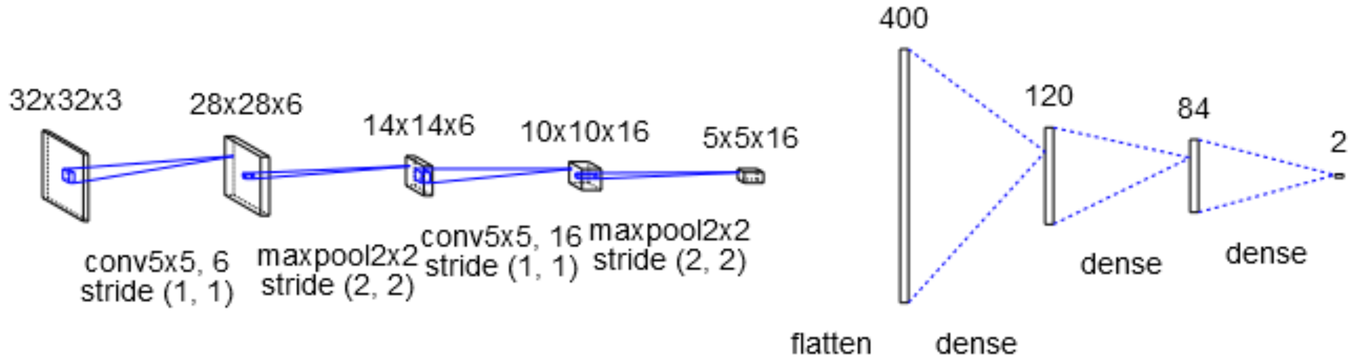
Figure 1: Colored MNIST

- Assign a binary label to the image based on the digits.
- Flip label with 25% probability.
- Color the image either red or green according to its possibly flipped label.
- Flip the color with a probability e that depends on the environment.

Preliminary Stage — LeNet

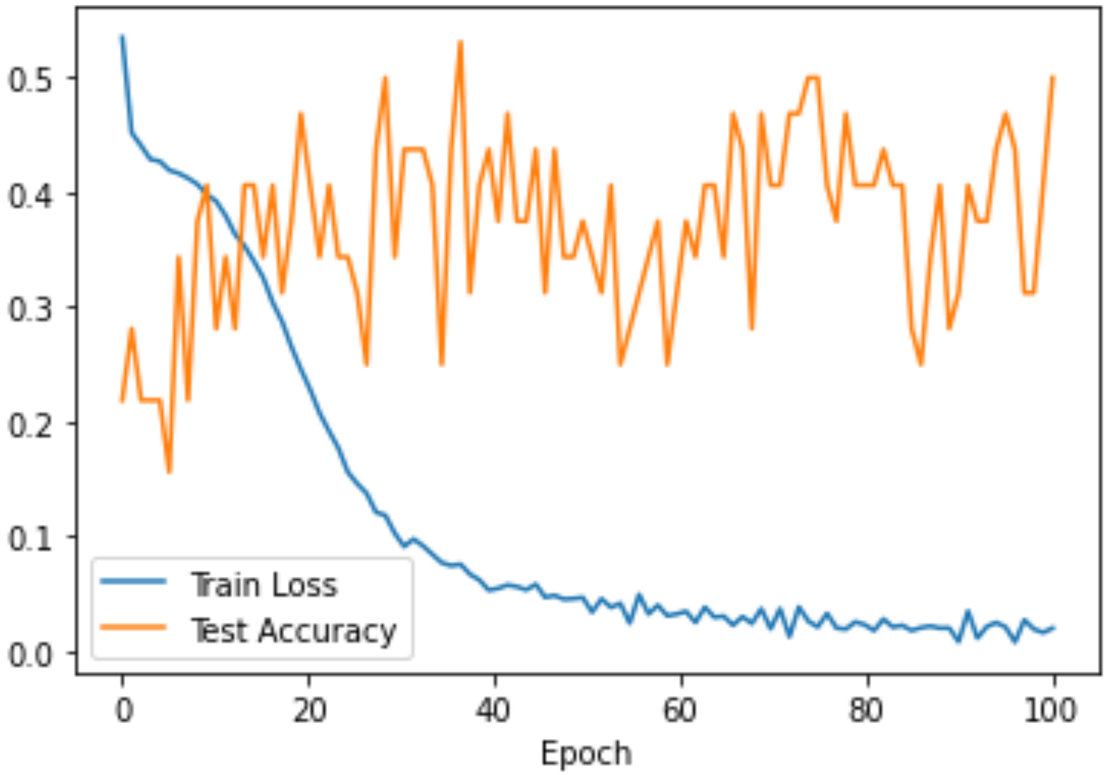
Standard LeNet Recurrence

	Raw LeNet	Modified LeNet
Input Size	(1,32,32)	(3,32,32)
Output Size	(10,)	(2,)



LeNet

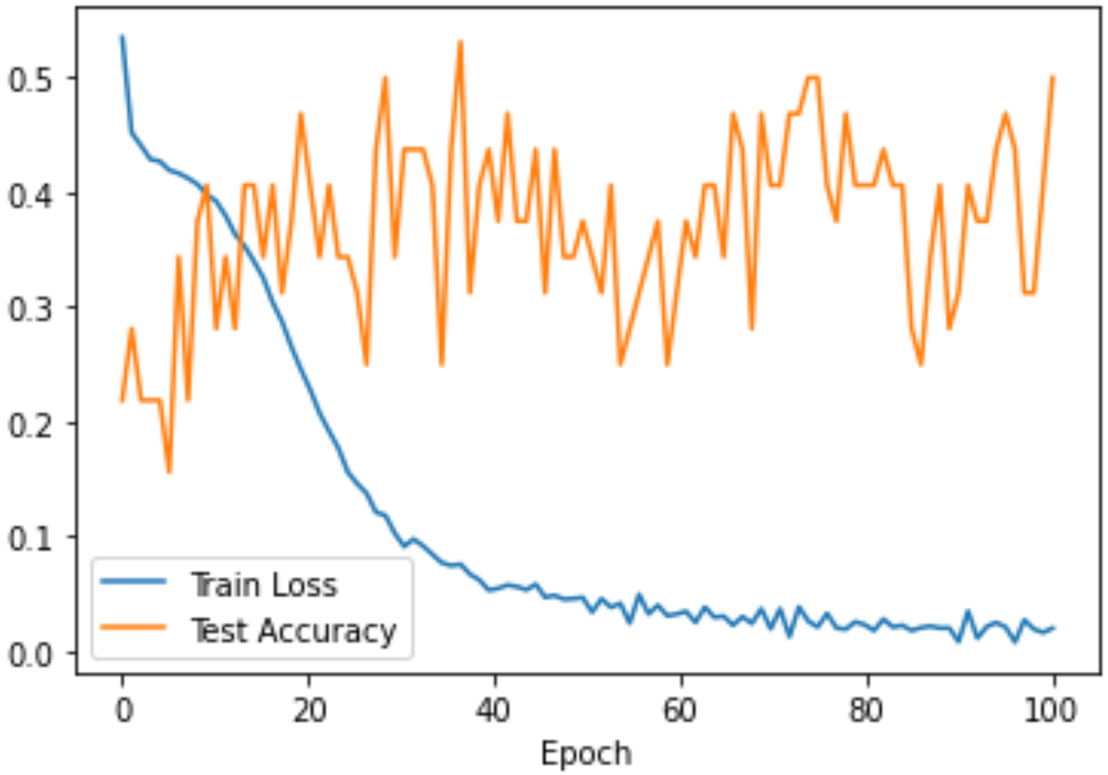
Standard LeNet



- Average Test Accuracy = 38%
- The accuracy curve vibrates violently.
- Showing no signs of convergence.
- Much Higher than MLP 10% (the unoptimized one implemented on task 3)
- But still far from enough.

LeNet

Standard LeNet

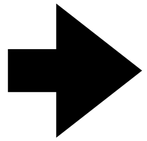


- We Need Data Augmentation

LeNet

Data Augmentation

- We Need Data Augmentation



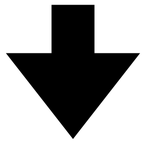
- Traverse the image (idx from 1 to 20000)
- Randomly flip color at the possibility of 20%
- Append them to the training dataset
- Repeat operations above for 10 times

Data Augmentation	Size	Label Flipped	Color Flipped	Average Accuracy
NO	20000	25%	20%	38.4%
YES	200000	25%	15%	56.2%

LeNet

Data Augmentation

Data Augmentation	Size	Label Flipped	Color Flipped	Average Accuracy
NO	20000	25%	20%	38.4%
YES	200000	25%	15%	56.2%

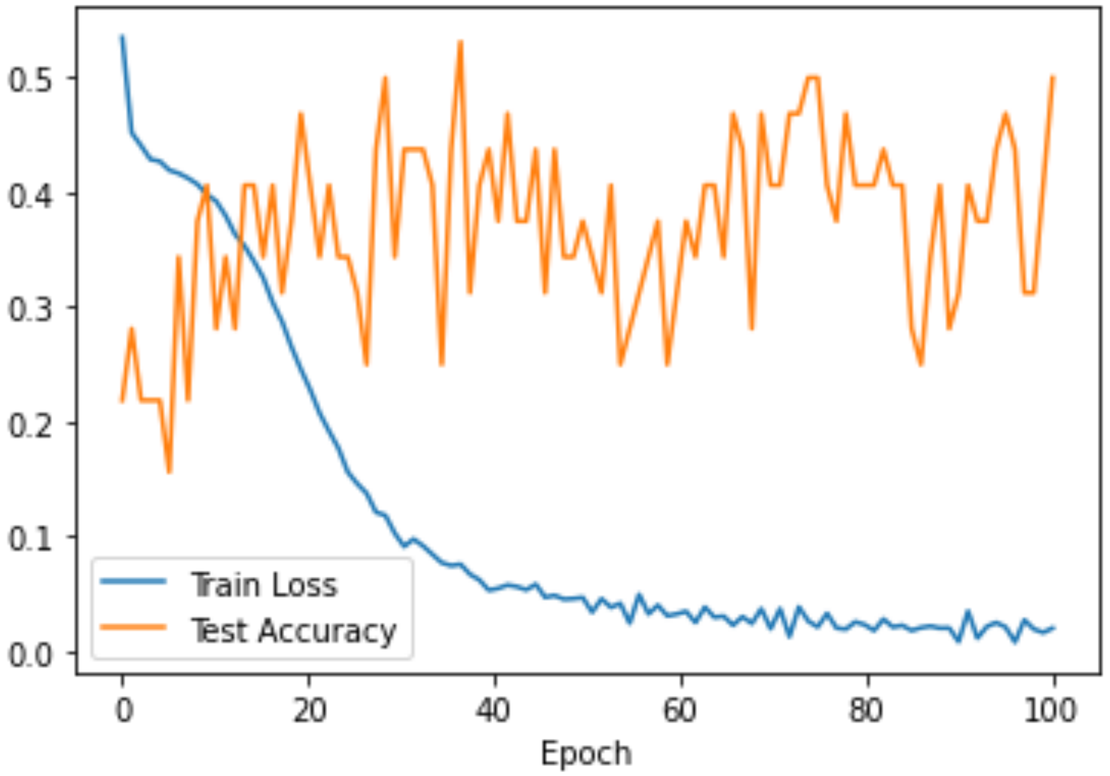


- Every image has shown up for 10 times and they are most likely to have different color.
- Still maintain the features previous raw dataset holds.

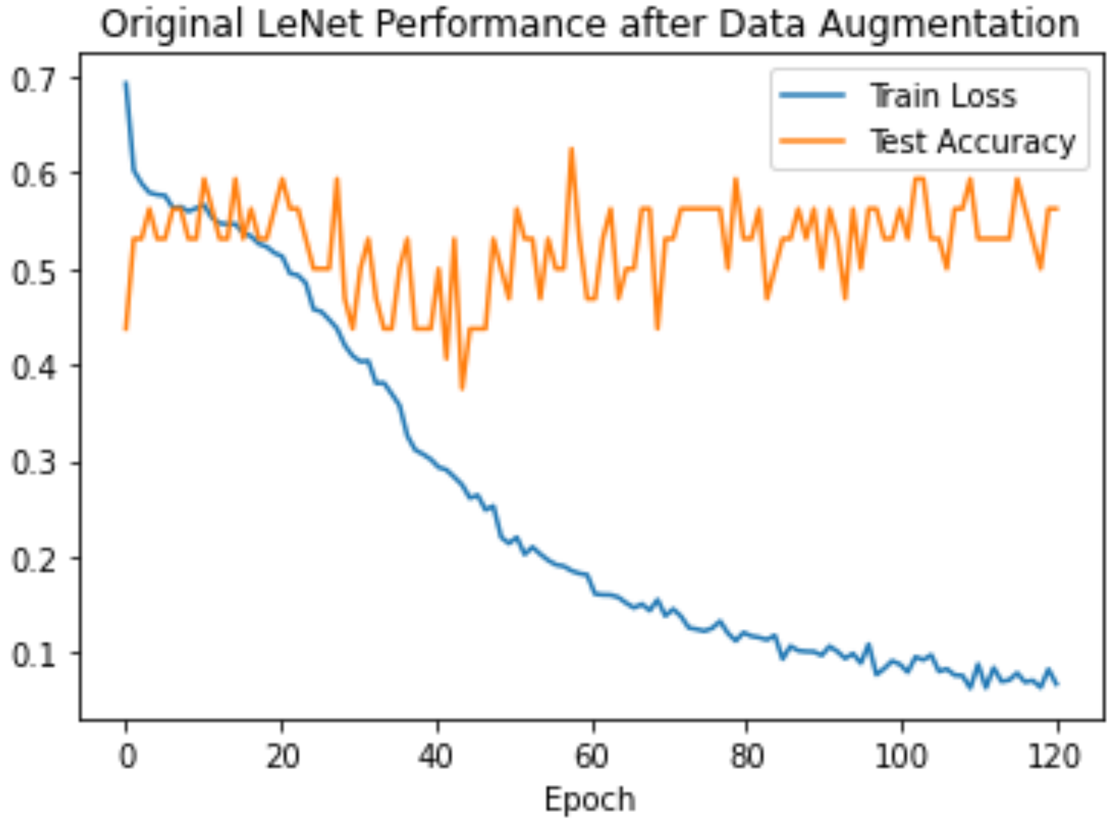
LeNet

LeNet trained on augmented data

Data Augmentation	Average Test Accuracy
No	38.4%
Yes	56.2%



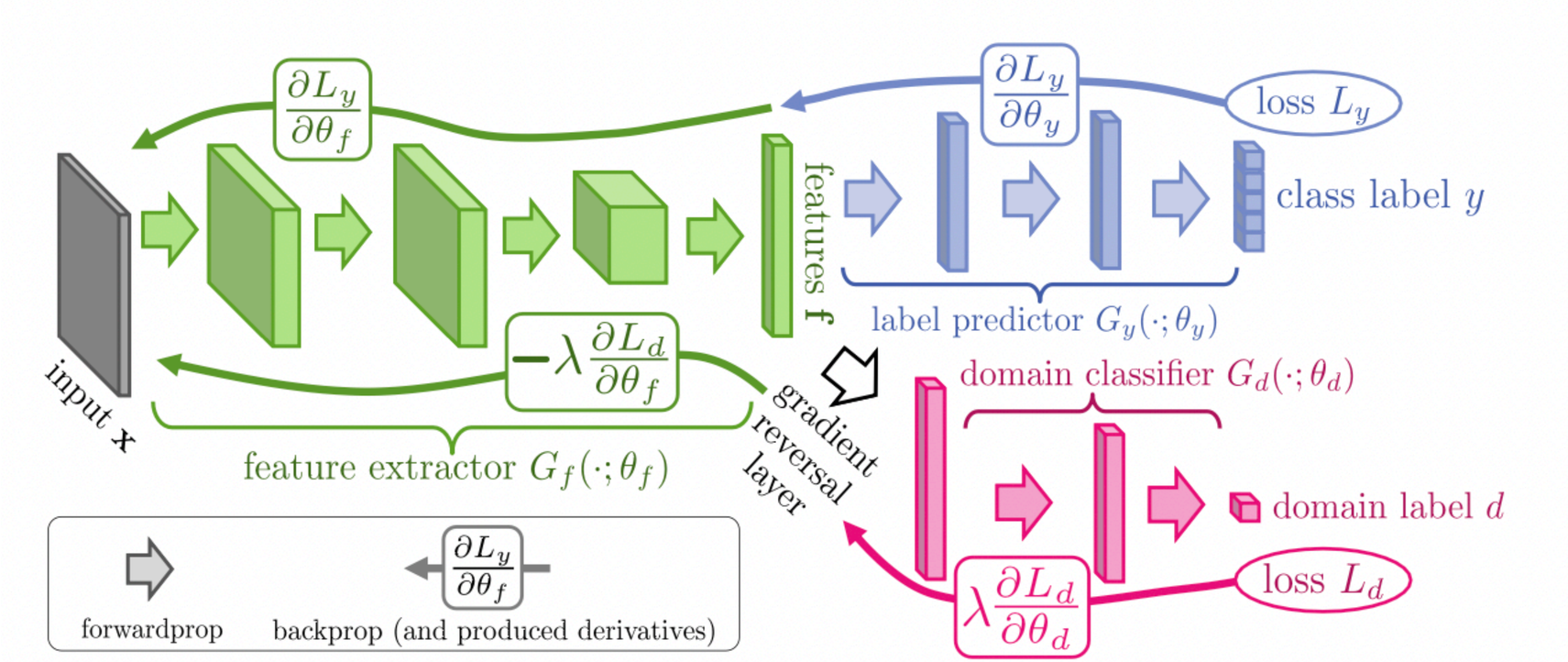
LeNet Performance without data augmentation



LeNet Performance with data augmentation

Domain-Adversarial Training of Neural Networks

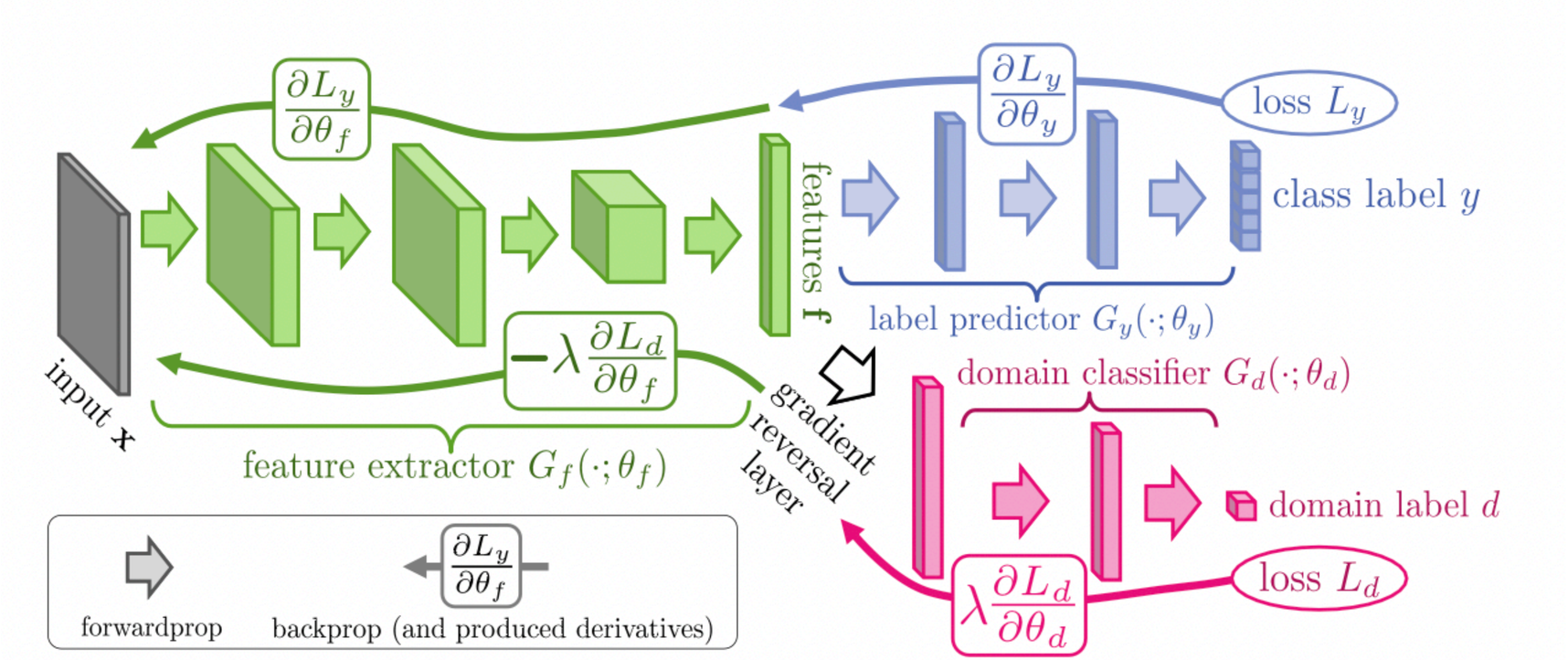
Overview of DANN



- A classical algorithm in the Adversarial Transfer Learning field.
- Ganin, Yaroslav et al. firstly introduce adversarial learning theory into transfer learning.
- **Domain Adaptation** is an important branch in transfer learning

Domain-Adversarial Training of Neural Networks

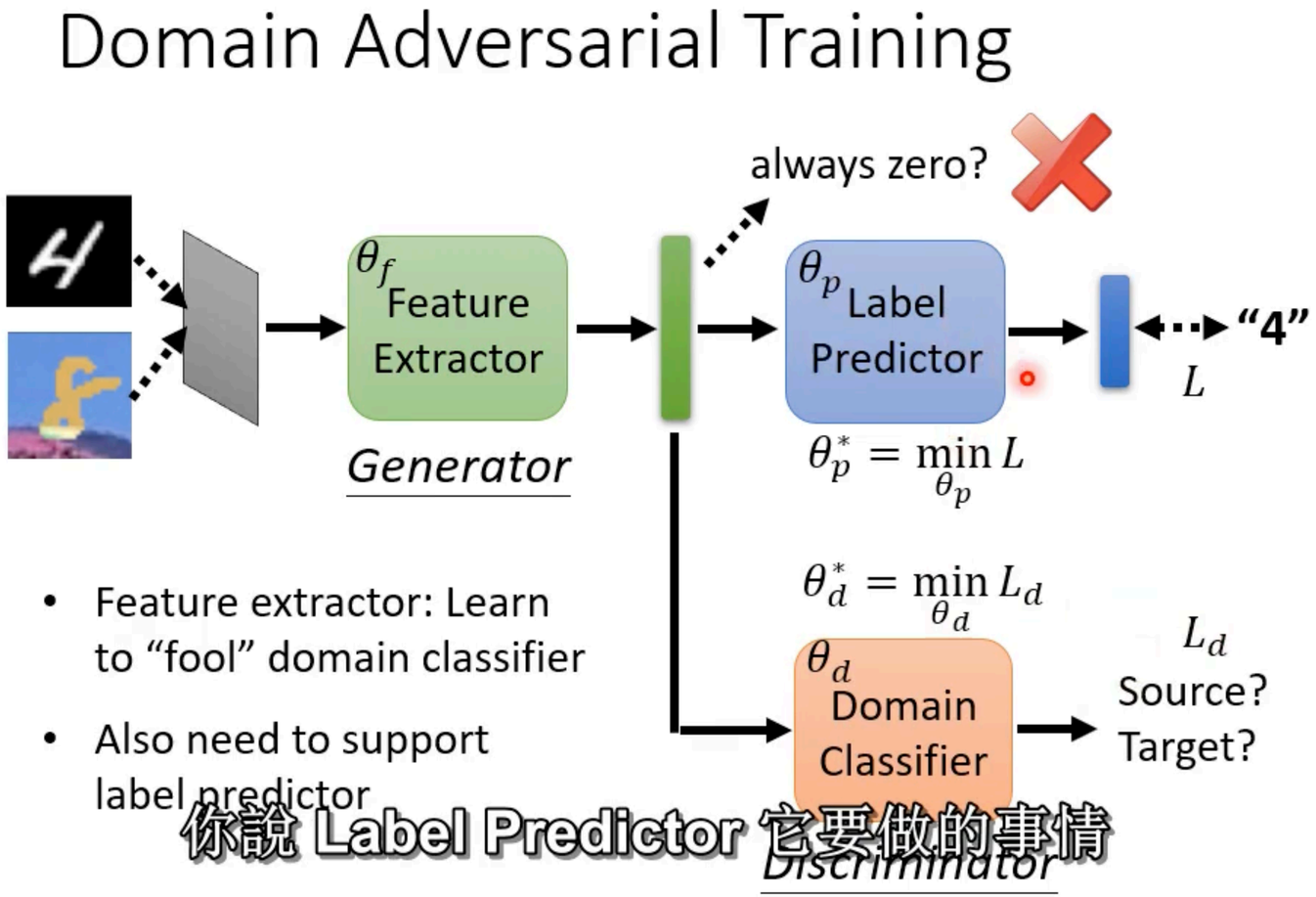
Overview of DANN



- Map source domain and target domain into the same eigenspace
- Find a certain distance measuring principle to minimize the distance between the mapped datasets.
- Directly use the label classifier trained on the source domain to classify data on the target domain

Domain-Adversarial Training of Neural Networks

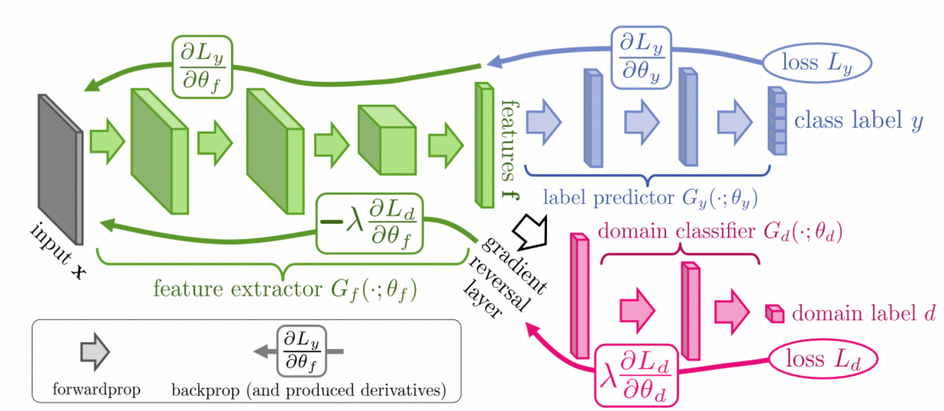
Recurrence of DANN



- **Feature Extractor:** Map data into an eigenspace so that while classifier is able to judge source domain data's labels, domain classifier cannot tell data apart.
- **Label Classifier:** Classify the data from source domain and acquire as high accuracy as possible
- **Domain Classifier:** Classify the data in the eigenspace and try its best to tell them apart.

Domain-Adversarial Training of Neural Networks

Theoretical Basis (Ben-David et al.(2006,2010))



$$R_S(\eta) + \beta + \hat{d_{\mathrm{H}}}(S,T)$$

$$\downarrow$$

$$R_S(\eta) + \hat{d_{\mathrm{H}}}(S,T)$$

Given two distribution: D_S^X D_T^X and a classification function H the $H_{divergence}$ can be written as :

$$d_H(D_S^x, D_T^x) = 2sup_{\eta \in H} \left| Pr_{x \sim D_S^x}[\eta(x) = 1] - Pr_{x \sim D_T^x}[\eta(x) = 1] \right|$$

$$\hat{d_H}(S,T) = 2\left(1 - min_{\eta \in H}\left[\frac{1}{n}\sum_{i=1}^{n} I[\eta(x_i) = 0] + \frac{1}{n'}\sum_{i=n+1}^{N} I[\eta(x_i) = 1]\right]\right)$$

$1-n$ come from the source domain and $n-N$ come from the target domain, i.e:

$$U = \{(x_i, 0)\}_{i=1}^{n} \cup \{(x_i, 1)\}_{i=n+1}^{N}$$

$$\hat{d_{\mathrm{A}}} = 2(1 - 2\epsilon)$$

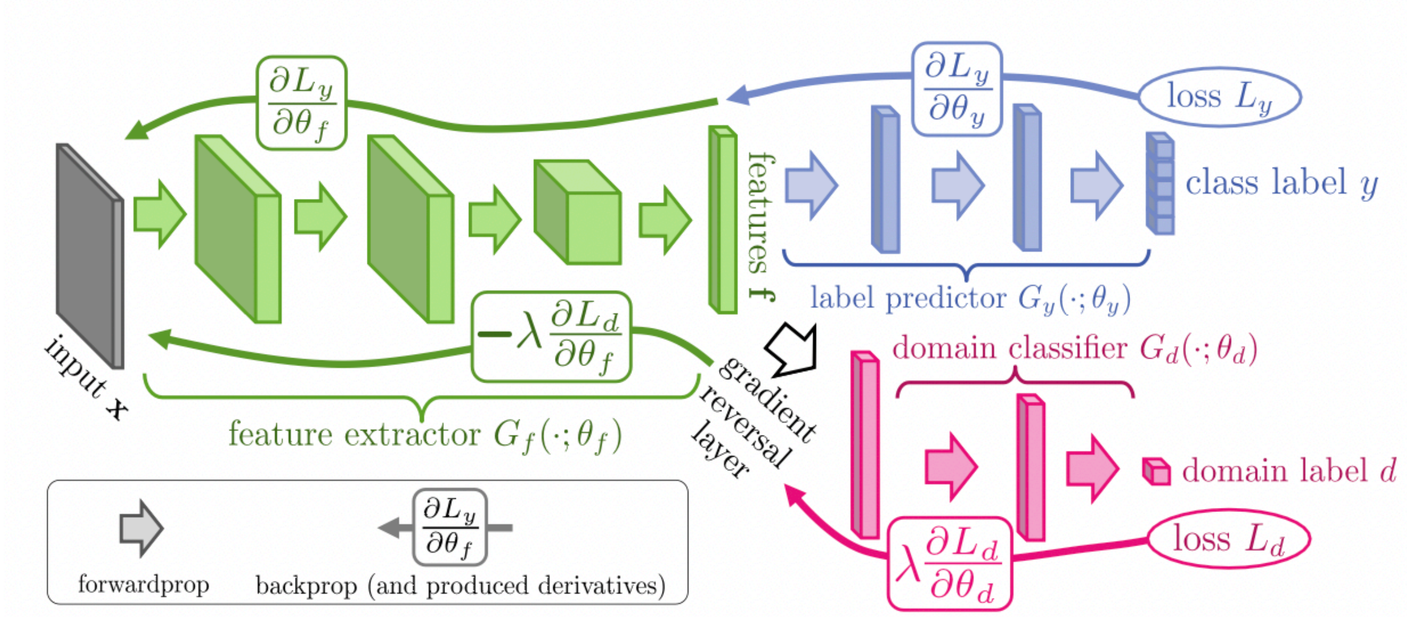
Meanwhile, Ben-David proved that $d_H(D_S^x, D_T^x)$ is the maximum of $\hat{d_H}(S,T) + O(d) + O(N)$. d represents the dimensionality of H 's VC dimension. So the risk on target domain satisfies:

$$R_{D_T}(\eta) \leq R_S(\eta) + \sqrt{\frac{4}{n}(dlog\frac{2en}{d} + log\frac{4}{\delta})} + \hat{d_H}(S,T) + 4\sqrt{\frac{1}{n}(dlog\frac{2n}{d} + log\frac{4}{\delta})} + \beta$$

$$\beta \geq inf_{\eta * \in \mathrm{H}}[R_{D_S}(\eta *) + D_{D_T}(\eta *)]$$

Domain-Adversarial Training of Neural Networks

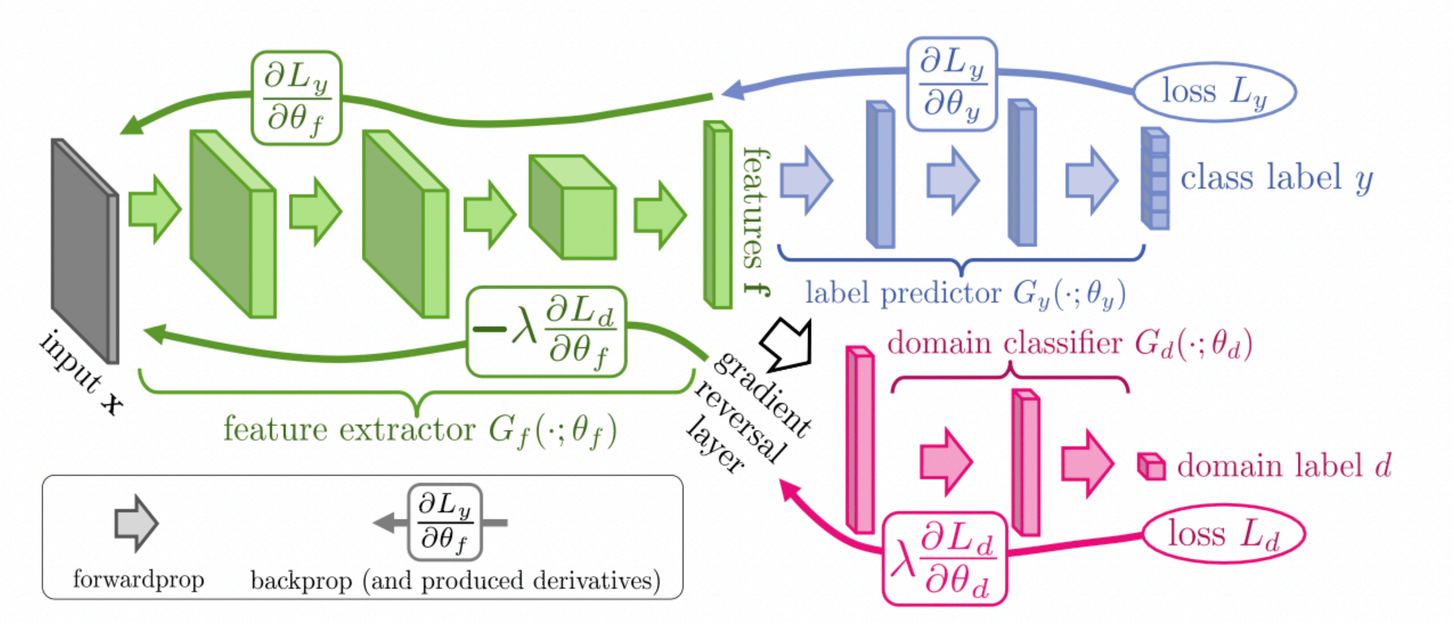
DANN Architecture



- Deep feature extractor and deep label predicator together form a standard feed-forward architecture
- Unsupervised domain adaptation is achieved by adding a domain classifier connected to the feature extractor via a GRL
- GRL ensures the feature distributions over the two domains are made similar(as indistinguishable as possible for the domain classifier), thus resulting in the domain-invariant features.

Domain-Adversarial Training of Neural Networks

DANN Architecture



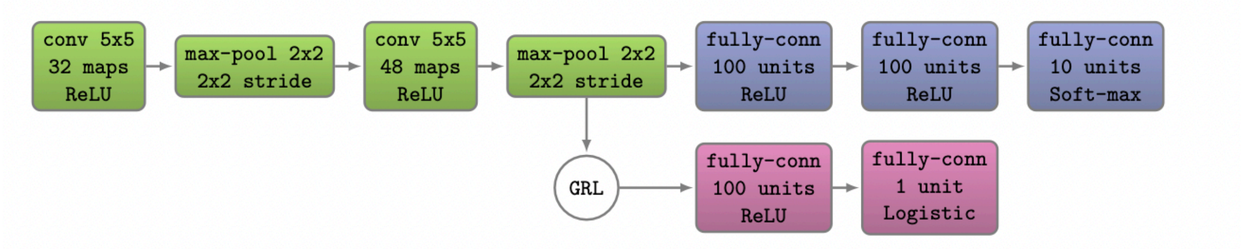

```python
from torch.autograd import Function

class GRL(Function):
    @staticmethod
    def forward(ctx, x, alpha):
        ctx.alpha = alpha
        return x.view_as(x)

    @staticmethod
    def backward(ctx, grad_output):
        output = grad_output.neg() * ctx.alpha
        return output, None
```

Domain-Adversarial Training of Neural Networks

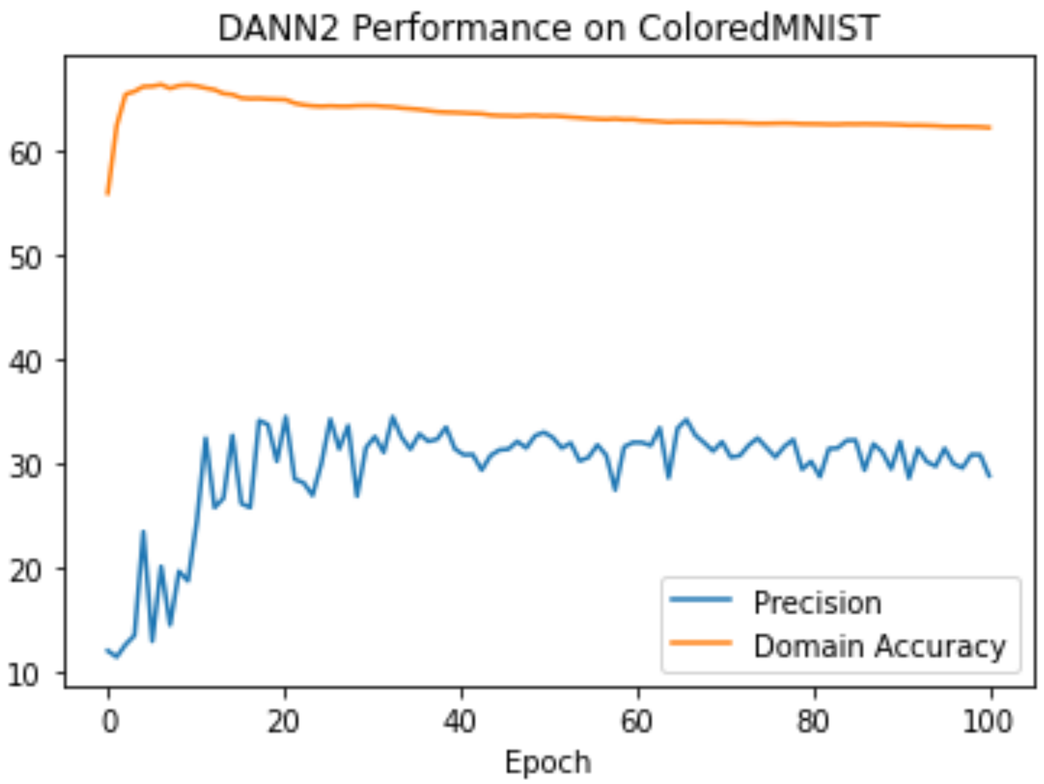
DANN Recurrence

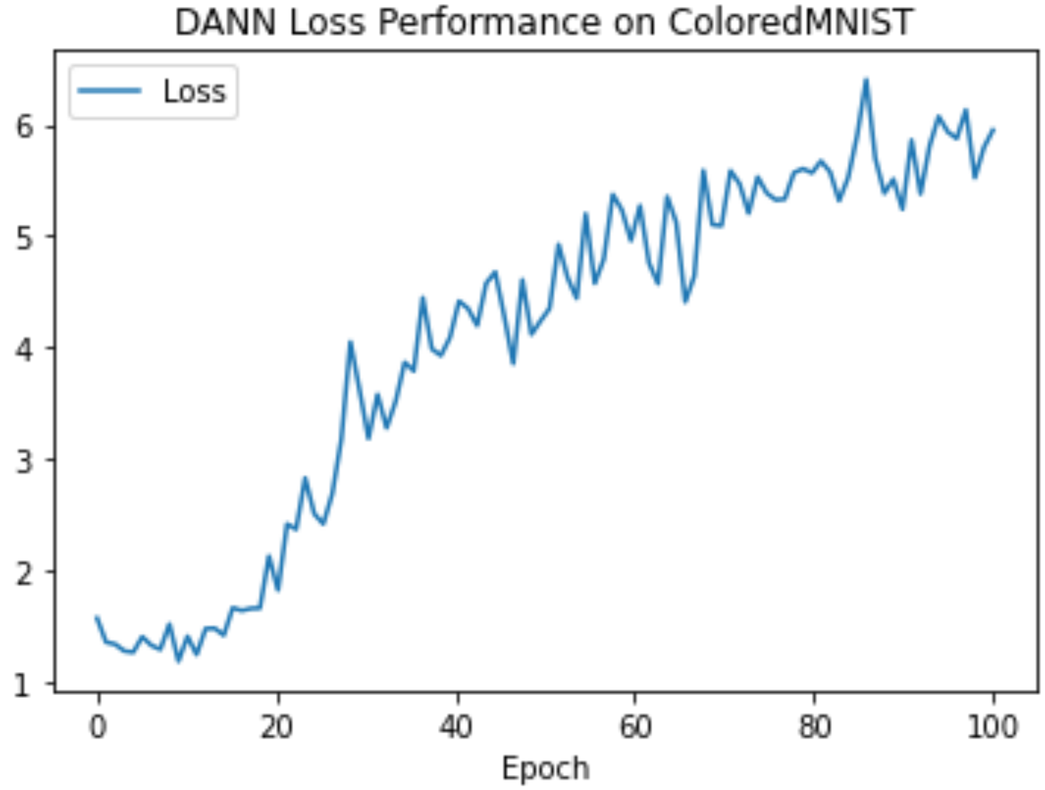



```python
class DANN(nn.Module):
    def __init__(self,num_classes=10):
        super(DANN,self).__init__()
        self.features=nn.Sequential(
            nn.Conv2d(3,32,5),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
            nn.Conv2d(32,48,5),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2),
        )
        self.avgpool=nn.AdaptiveAvgPool2d((5,5))
        self.task_classifier=nn.Sequential(
            nn.Linear(48*5*5,100),
            nn.ReLU(inplace=True),
            nn.Linear(100,100),
            nn.ReLU(inplace=True),
            nn.Linear(100,num_classes)
        )
        self.domain_classifier=nn.Sequential(
            nn.Linear(48*5*5,100),
            nn.ReLU(inplace=True),
            nn.Linear(100,2)
        )
        self.GRL=GRL()
    def forward(self,x,alpha):
        x = x.expand(x.data.shape[0], 3, image_size,image_size)
        x=self.features(x)
        x=self.avgpool(x)
        x=torch.flatten(x,1)
        task_predict=self.task_classifier(x)
        x=GRL.apply(x,alpha)
        domain_predict=self.domain_classifier(x)
        return task_predict,domain_predict
```

Domain-Adversarial Training of Neural Networks

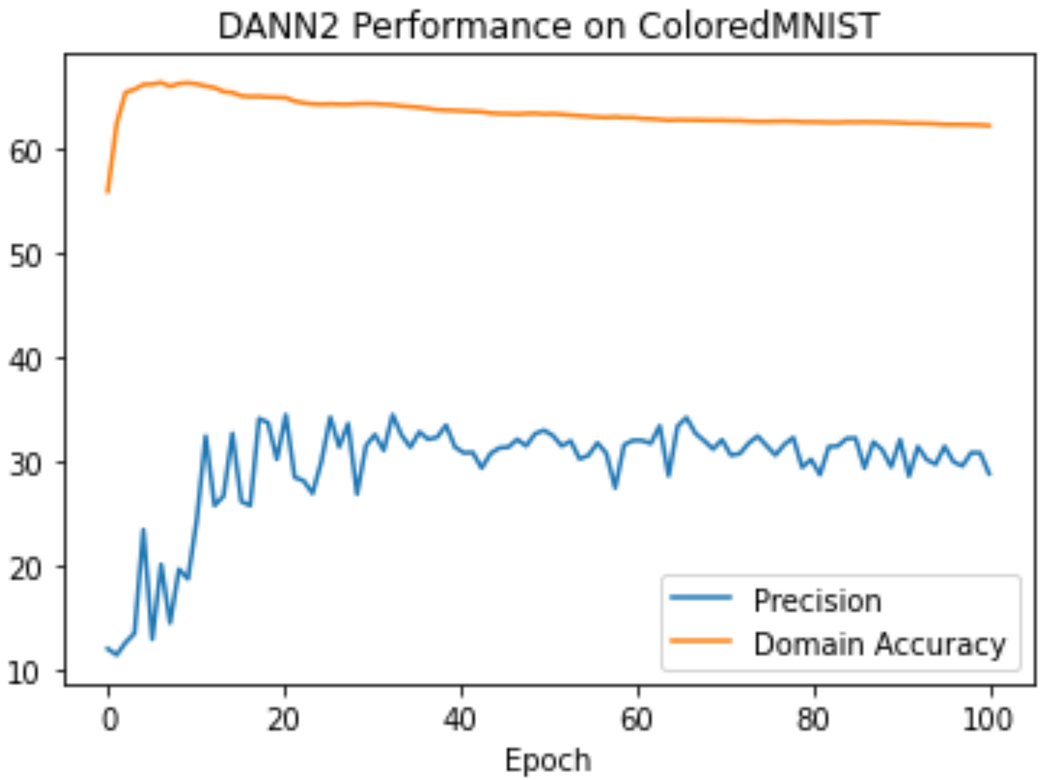
DANN Recurrence





Domain-Adversarial Training of Neural Networks

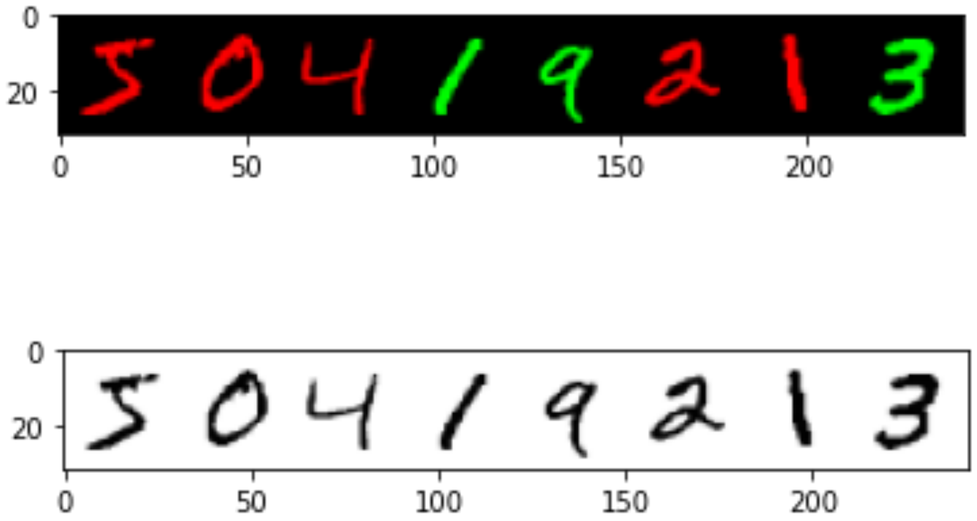
DANN Recurrence



- Feature extractor still has learned much from color information due to the comparatively rather small color-flipping rate on train1 and train 2
- That also leads to the low domain accuracy
- Change training way.

Domain-Adversarial Training of Neural Networks

DANN Recurrence — Preparing MNIST-M



- Assign a binary label to the image based on the digits.
- Flip label with 25% probability.
- Color the image either red or green according to its possibly flipped label.
- Not flip the color

Domain-Adversarial Training of Neural Networks

DANN Recurrence — Preparing MNIST-M

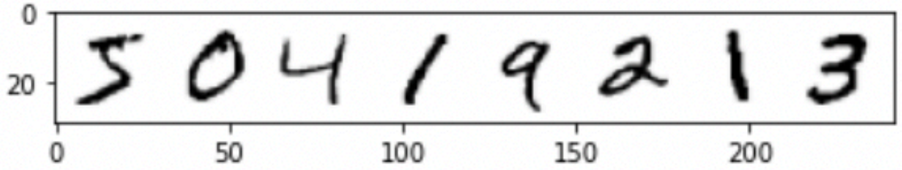


Figure 10: Source Domain

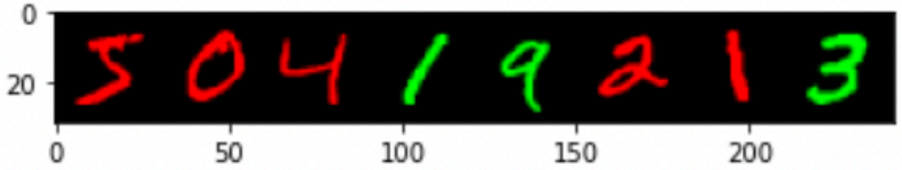


Figure 11: Target Domain

Training Method	source-only training accuracy	direct transferring accuracy	Color Flipped	Average Accuracy	Domain Accuracy
Colored MNIST	89.4%	13.67%	90.03%	31.02%	65.50
MNIST-M	73%	72.3%	30.3050%	50.41%	100%

Domain-Adversarial Training of Neural Networks

DANN Recurrence — Preparing MNIST-M

Training Method	source-only training accuracy	direct transferring accuracy	Color Flipped	Average Accuracy	Domain Accuracy
Colored MNIST	89.4%	13.67%	90.03%	31.02%	65.50
MNIST-M	73%	72.3%	30.3050%	50.41%	100%

Table 3: Through this way, source-only training accuracy is about 73%(considering the theoretical maximum is 75%, this CNN model is very promising), direct transferring accuracy is 72.3%, direct training accuracy is 30.3050%, and finally DANN accuracy is 50.41%

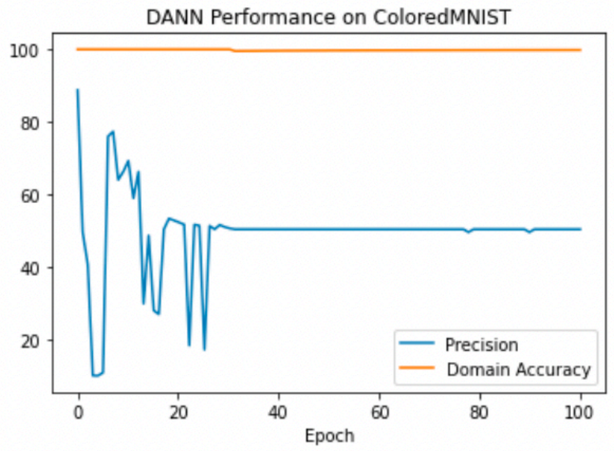


Figure 12: Precision and Domain Accuracy

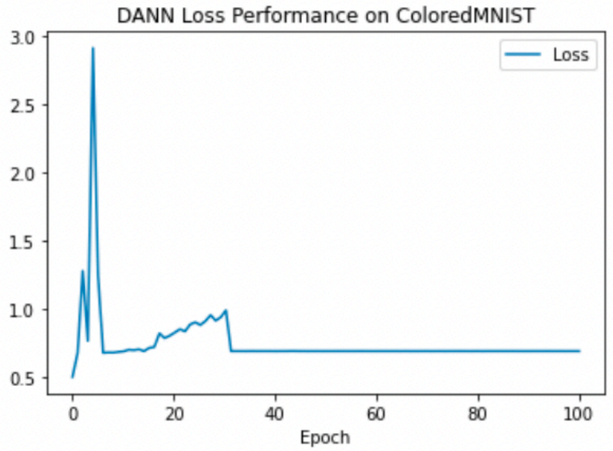
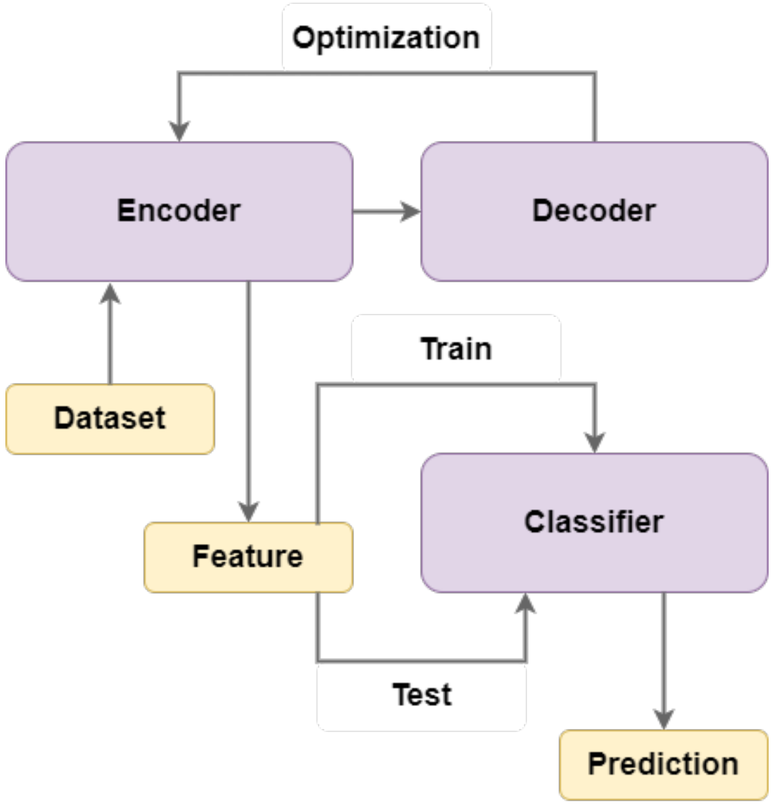


Figure 13: Loss

Further Exploration

Research on Ensemble Learning Algorithms' performances



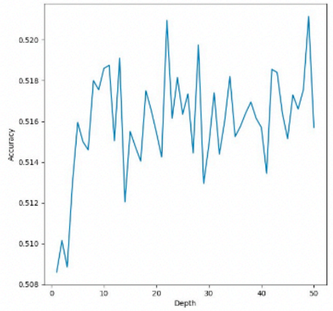


Figure 16: AE + RF

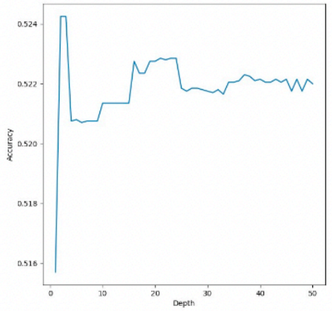


Figure 17: AE + AdaBoost

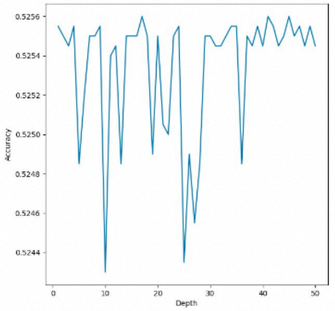


Figure 18: AE + GradientBoost

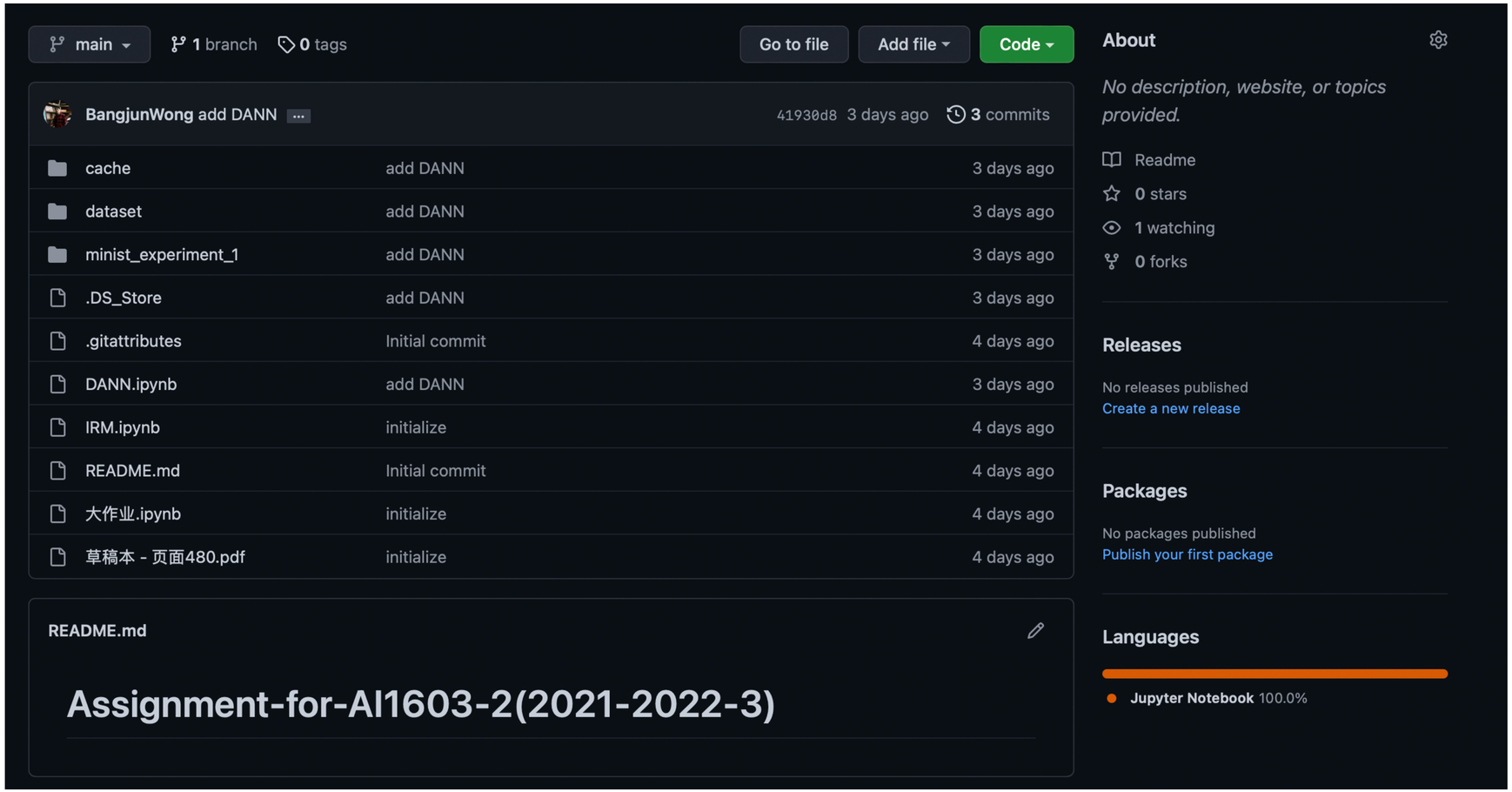
Classifier	Random Forest	AdaBoost	Gradient Boost	Boost Average
Average Accuracy	51.6%	52.2%	52.50%	52.35%
Peak	52.1%	52.4%	52.56%	50.48%
Valley	51.2%	52.1%	52.44%	52.27%
Stability	*Violent*	*Gentle*	*Steady*	*Steady*
Amplitude	0.9%	0.3%	0.12%	0.21%

Table 4:

Relative research details can be found in my assignment report.

Code Repository

All the source code, plots and report will be public on Github.



Later I'll push all the sorted files to it.

Reference

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[2] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[4] Cryan. Cryan's experiments on domain adaptation. [EB/OL]. `https://zhuanlan.zhihu.com/p/126185010` Accessed July 10, 2022.

[5] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021.

[6] Giambattista Parascandolo, Alexander Neitz, Antonio Orvieto, Luigi Gresele, and Bernhard Schölkopf. Learning explanations that are hard to vary. *arXiv preprint arXiv:2009.00329*, 2020.

[7] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *European Conference on Computer Vision*, pages 124–140. Springer, 2020.