Report of Brain-Inspired Computing Final Project

Letian Yang
moekid101@sjtu.edu.cn

Runkun Chen
phtwrccc@sjtu.edu.cn

Haoyu Zheng
langanzheng@sjtu.edu.cn

1. Introduction

The given dataset is *Colored-MNIST*. All our work are done on the *Colored-MNIST* dataset as an attempt to understand *causal inference* and *out-of-distribution generalization*. The generation of *Colored-MNIST* could be described by Fig.(1). The details are explained as follows:

(1) Determination of label using digit. It is assumed that $(\tilde{X}, \tilde{Y})$ is sampled from the original MNIST dataset $\left\{\left(\tilde{x}^{(i)}, \tilde{y}^{(i)}\right) | i \in \{1, 2, ..., N_0\}\right\}$ where the label $\tilde{Y}$ is binary-valued, 0 indicating a digit ≤ 4 while 1 indicating a digit ≥ 5. For given digit $\tilde{Y}$, the final binary label Y is determined at random by

$$\mathbf{Pr}\left[Y = \tilde{y} | \tilde{Y} = \tilde{y}\right] = 0.75 \tag{1}$$

(2) Generating color with label and environment. Three environments are presented as three datasets, $E \in \{e_1, e_2, e_3\}$. The generation of color Z follows

$$\begin{pmatrix} \mathbf{Pr}\left[Z = y | Y = y, E = e_1\right] \\ \mathbf{Pr}\left[Z = y | Y = y, E = e_2\right] \\ \mathbf{Pr}\left[Z = y | Y = y, E = e_3\right] \end{pmatrix} = \begin{pmatrix} 0.9 \\ 0.8 \\ 0.1 \end{pmatrix} \tag{2}$$

(3) Shading MNIST Images. The sampled MNIST image $\tilde{X}$ is shaded into X by the binary random variable Z. X is red if $Z = 1$ and X is green if $Z = 0$. Then the generated tuple (X_e, Y_e) is constructed as the new dataset *Colored-MNIST*.

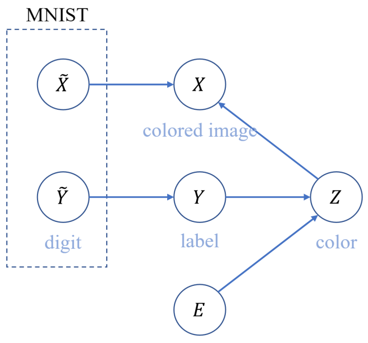


Figure 1. Generation of *Colored-MNIST*.

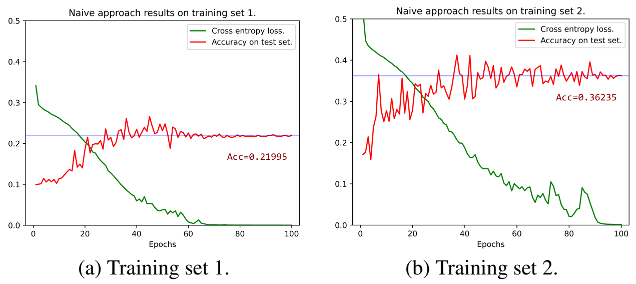
The dataset generation suggests that although the determinant feature of labels are digits, there exists stronger correlation between labels and colors. When environment changes, such correlation fails to generalize. Therefore, to perform well on the test dataset requires that our adopted algorithm have sufficient knowledge of *OoD generalization*, clearly identifying the *causal feature* being shapes instead of colors (defined as *spurious features*).

The majority of this report has been covered in the our group's presentation. Some supplementary ideas and details are highlighted in dark red in this report (similar to this sentence).

2. Baseline

The baseline method, as the term suggests, ignores all causality and directly train CNNs on the training sets.

At this stage, we claim that models should be trained until full convergence in most cases. Note that we are testing the neural network on a correlation remarkably different from the one it is trained upon. Therefore, small training steps potentially yield substantial performance variations on the test set, and the test set accuracy might even be dependent on the order in which the training set is loaded (as the last few batches may dominant model performance). Consequently, reliable analysis should be grounded on models that have achieved full convergence.



(a) Training set 1. (b) Training set 2.

Figure 2. Baseline performance.

We experiment on both of the training sets and test its accuracy on the test set, achieving 22.0% and 36.2% accuracy respectively (notably, both models are trained 100 epochs to ensure full convergence). Results on accuracy align with straightforward probabilistic calculation based on Eq.(2), and thus are not further elaborated on.

3. Task 1: Twin Model

In the first task, we assume the context labels are available for causal inference. Each training samples is composed of a 3-tuple $\left(x^{(i)}, y^{(i)}, z^{(i)}\right)$, where $z^{(i)}$ represents the context label (color in ColoredMNIST). In the testing phase, we assume no knowledge of the context label.

In this section, we begin with causal inference, or more specifically, *do* calculus using *backdoor criterion*. Basing our model on a simplified causal diagram as shown in Fig.(3), our ultimate goal is formulated as

$$\mathbf{Pr}\left[Y|do(X)\right] = \sum_{z\in\{0,1\}} \mathbf{Pr}\left[Y|X, Z=z\right]\mathbf{Pr}\left[Z=z\right] \quad (3)$$

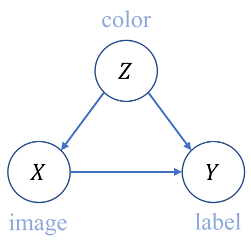


Figure 3. Causal diagram of Twin Model.

Inspired by Eq.(3), we propose a new approach by modelling probability on different context labels separately, named *Twin Model*. While the baseline models fail at modelling the correlation $\mathbf{Pr}\left[Y|X\right]$ due to the spurious correlation introduced by Z, we instead train two sub-models in parallel, using only red (or green) training samples, so that each individual model would learn the correlation $\mathbf{Pr}\left[Y|X, Z=z\right]$ conditioned on each context label instead.

3.1. Implementation and Experiments

Specifically, we formulate our idea by Eq.(4). The twin model output is the weighted average of that from Model_0 and Model_1, where Model_i is trained only on the training samples subject to $Z=i$.

$$\begin{aligned}\text{TwinModel}(Y|X) =&\mathbf{Pr}\left[z=0\right]\text{Model}_0(Y|X)+\\ &\mathbf{Pr}\left[z=1\right]\text{Model}_1(Y|X) \quad (4)\end{aligned}$$

In Eq.(4), $\mathbf{Pr}\left[z=0\right]$ and $\mathbf{Pr}\left[z=1\right]$ are obtained through statistical means, by counting the numbers of samples where $z=0$ or $z=1$. As our simplified causal diagram only accounts for context labels (Z) but not environment (E) per se, for Twin Model to work, we have to assume that $\mathbf{Pr}\left[z\right]$ remains relatively consistent in different environments.

We carry this idea into practice and conduct a series of experiments.

For the architecture of Model_0 and Model_1, we use a standard CNN inspired by LeNet. To test the viability of our method, we train our model with different model sizes (small model or large model), optimizers (SGD or Adam) and training parameters. Due to class imbalance in the training dataset, however, certain sets of parameters fail to converge in reasonable time. This effect is further discussed in 3.2.2.

In our experiment, we settle on training the model with an SGD optimizer at learning rate $=0.01$, weight decay $=10^{-4}$, momentum $=0.9$, for 100 epochs. The resulting loss and accuracy is reported in Fig.(4).

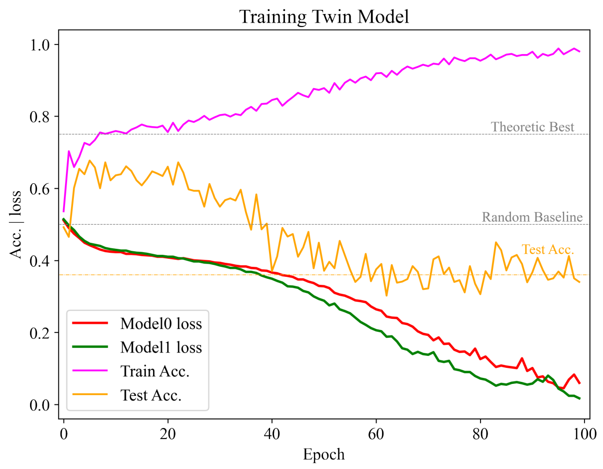


Figure 4. Progression of Twin Model training loss and model accuracy over 100 epochs.

As is shown in the figure, while the model performs relatively well in the first 40 epochs, as training proceeds, test accuracy falls to below random baseline despite decreasing loss. This can partly be attributed to overfitting: as the model's accuracy on training sets exceeds 75%, Twin Model increasingly learns false correlation from the training data. Overfitting is further worsened by class imbalance, to be discussed later.

On the other hand, as Model_i is directed to learn the correlation between image and label conditioned on $Z=i$, it is unable to predict images generated with different context $Z=1-i$ reliably. As a result, the model can output wildly different and unstable outputs when given an image with opposing color. To analyze this effect, we introduce *class imbalance* and *confidence loss* to better understand the causes.

3.1.1 Tranplanting to Other Architectures

We also transplant the parameters of trained twin model to *spiking neural networks* (SNN) and *memristor deep neural networks* (MDNN). We choose $\beta=0.9$ for SNN and ideal

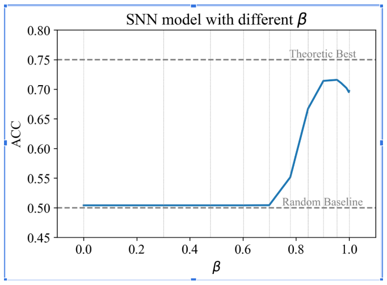


Figure 5. Accuracy of SNN twin model with different βs.

memristor. The output is demonstrated in Table.1.

Table 1. Results under different architectures.

Model	Twin Model	SNN	memristor
Test acc	68.295	70.895	61.22

We further our study on the SNN architecture by recording the performance of the *Twin Model* under different β values, the result of which is shown in Fig.(5). We see from the figure that when β falls in $[0.9, 0.97]$, the SNN model attains its peak performance.

3.2. Locating Problems: Class Imbalance and Confidence Loss

To analyze the reasons behind the observed shortcomings and explore new potential solutions, we pursue distinct avenues of investigation. In particular, we propose **Confidence Loss** to measure each sub-model's failure to generalize to different contexts, and introduce an analysis on the effects of **class imbalance**.

3.2.1 Confidence Loss

Unique to our architecture, each sub-model inside Twin Model is trained independently on separate data. As a result, each sub-model performs poorly on data with opposing context color, as they are unseen during training. The probability predictions the sub-models yield on opposing are indiscriminately off-centered and over-confident. More often than not, the sub-model produces more confident result on opposing color than the color it is trained on.

This leads to unstable mean prediction, and causes the model's best accuracy to differ substantially with each experiment. To analyze the effect, we introduce a custom metric, named Confidence Loss, to evaluate te over-confidence phenomenon. We define Confidence Loss for $\boldsymbol{p} \in \Delta^1$ as below.

$$\mathcal{L}_{\text{Conf}}(\boldsymbol{p}) = \text{CrossEnropy}(\boldsymbol{p}, \arg\max \boldsymbol{p}) \tag{5}$$

We evaluate the Confidence Loss on a new training process and report the results in Fig.(6). Specifically, we evaluated the red sub-model on red or green subset, and the green sub-model on red or green subset, respectively. The results show significant correlation between sub-models' confidence loss on opposing colors and the Twin Model's accuracy. In earlier epochs, each sub-model's Confidence Loss with regards to the color it is trained with manages to reduce rapidly (i.e. able to predict image labels), while Confidence Loss for opposing colors remains high; in later epochs, as Confidence Loss for opposing colors drop, the Twin Model's overall accuracy also decrease. Our results validate the observation of model over-confidence.

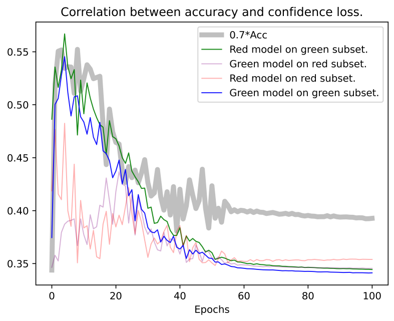
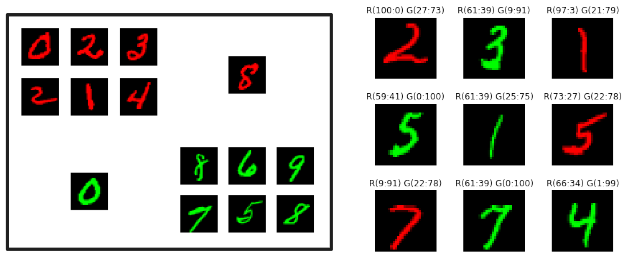


Figure 6. Correlation between confidence loss and accuracy. To be intuitive, we multiply the accuracy by 0.7 so that the curve approximates the blue and green line segments.

3.2.2 Class Imbalance

In our earlier experiments, preliminary results show that Twin Model is particularly prone to underfitting caused by class imbalance. Plainly put, the distribution of training samples of different labels Y is too imbalanced, as illustrated in Fig.(7), for the neural network to learn. In several attempts at training, after 20 epochs, the loss function fails to decrease in meaningful terms. This effect can be partly alleviated by adjusting the parameters and changing the optimizer to SGD, allowing the model to converge.



(a) Dataset Composition (b) Predictions

Figure 7. Left: an illustration of the imbalanced distribution of labels and color contexts in the training set. Right: some sample results from Twin Model. $\text{R}(100p : 100 - 100p)$ means the red sub-model gives a probability of p for $Y = 0$ and $1 - p$ for $Y = 1$.

In converged models, class imbalance manifests as a unique category of overfitting. Due to imbalance distribution among classes, the training process tends to exemplify false correlations introduced in the training sets. Such false correlation is learned by the model, and gradually decrease the model's accuracy to below random. One way to avoid overfitting due to class imbalance is simply *train less epochs*; other potential solution to this problem is explored in the next section.

3.3. Lesson From Twin Model: Let's Go Beyond Causal Inference

Up to this point, we have scrutinized two reasons for the failure of our proposed model based on the assumed causal diagram Fig.(3), which implies the following statements:

- The marginal probability $\mathbf{Pr}\left[Z=z\right]$ is invariant over training set and test set.
- There exists stable correlation $\mathbf{Pr}\left[Y|X,Z=z\right]$ in all environments.

However, with the knowledge of dataset generation process (Fig.(1)), the conclusion can be readily drawn that **both of the statements are wrong**. However, all our efforts paid into training an effective model with proper conjectures ended in failure. Let's simply jump into our conclusion.

In order to rectify the second statement, it is imperative that we explicitly include the environment E as a random variable in the causal diagram and model $\mathbf{Pr}\left[Y|X,Z,E\right]$ instead. However, it suggests that purely causal methods are **entirely unfeasible** to solve the problem presented by *Colored-MNIST*, because the environment of test set (e_3) is always unseen, and there is no way we can learn an invisible conditional probability $\mathbf{Pr}\left[Y|X,Z,E=e_3\right]$ from observed dataset.

The preceding logic helps us reach the conclusion that causal inference is inefficient, and we should pursue a different approach in learning the *Colored-MNIST* dataset. The remaining sections of this report are contributed to describe our efforts in doing so.

4. Task 2: Balanced Sampling

In task 2, we no longer have access to context labels in training set. Instead, we are exposed to nothing more than $\left\{\left(x_e^{(i)},y_e^{(i)}\right)|i=1,2,...,N_e\right\}\sim\mathcal{D}^e$ for every training environment $e\in\mathcal{E}_{\text{train}}$, while the context label $z_e^{(i)}$ are unobserved.

Previous researches are mostly focused on designing robust learning algorithms immune to spurious correlations varying across environments. the representatives include *invariant risk minimization* [1] and *group DRO* [2].

However, after investigating into and conducting experiments on several popular ideas, I find the performance of most methods are dependent on the spurious features themselves. This inspired me to take a different (in fact, orthogonal) point of view, that is, change the datasets.

As a result of extensive literature review, I drew inspiration from Xinyi Wang et al. [3].

4.1. Causal Balancing for Domain Generalization

In this paper, the author embraced the perspective of modifying datasets, and proposed a two-step method to solve the spurious correlation problem based on the theory of *balanced distribution.*

Def Balanced Distrbution: A balanced distribution is written as $p^B(X,Y,Z)=\mathbf{Pr}\left[X|Y,Z\right]p^B(Z)p^B(Y)$ where $p^B(Y)=U\{1,2,...,m\}$ and $Y\perp\!\!\!\perp_B Z$.

Notably, the *Colored-MNIST* dataset features $\mathbf{Pr}\left[Z=z\right]=0.5$ for any environment while the conditional probability $\mathbf{Pr}\left[Z=z|Y=z\right]$ varies among environments as Eq.(2). Therefore, if we can balance the distribution of *Colored-MNIST*, our neural networks could simply ignore the irrelevant feature Z and model the correlation between label Y and input X based on the causal feature $\tilde{Y}$. Here, conditional independence shifts our causal diagram as is shown in Fig.(8).

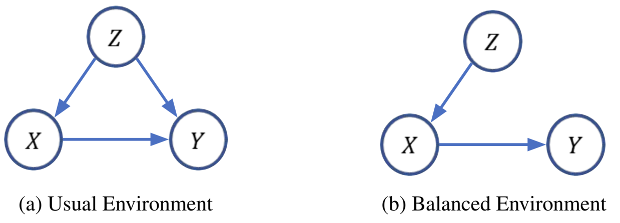


Figure 8. Causal diagrams for different environments.

Def Balancing Score: A balancing score $b(Z)$ is a function of Z such that $Z\perp\!\!\!\perp Y|b(Z)$.

We interpret the definition intuitively as follows. From the perspective of utilizing the definition, balancing score is defined as a condition that yields independence, or equivalently, balance. However, from the perspective of judging whether a function is a balancing score, it is straightforward enough that $b(Z)$ is a balancing score if and only if

$$\mathbf{Pr}\left[Y|Z\right]=\mathbf{Pr}\left[Y|b(Z)\right] \tag{6}$$

Therefore, we can conclude that the finest balancing score is Z itself (trivial) while the coarsest one is $b_0(Z)=\mathbf{Pr}\left[Y=1|Z\right]$ (assuming binary Y and Z, as is the case for *Colored-MNIST*). No balancing score is coarser than $b_0(Z)$ because such function yields $b(z_1)\neq b(z_2)$ while $b_0(z_1)=b_0(z_2)$, which means $\mathbf{Pr}\left[Y=1|z_1\right]\neq\mathbf{Pr}\left[Y=1|z_2\right]$, and thus conditional dependency between Y and Z. On the other hand, $b_0(Z)$ is a balancing score because

$$\begin{aligned} b_0(Z) =& \mathbf{E}\left[b_0(Z)|b_0(Z)\right] \\ =& \sum_z \mathbf{Pr}\left[Y=1|Z=z\right]\mathbf{Pr}\left[Z=z|b_0(Z)\right]\mathrm{d}z \\ =& \mathbf{Pr}\left[Y=1|b_0(Z)\right] \end{aligned}$$

The author adopted the vector version $s(Z) = [\mathbf{Pr}\left[Y=y|Z\right]]_1^m$ of $b_0(Z)$ as his choice of balancing score and generalized his approach to non-binary Ys. The author provided the proof of $s(Z)$ being the coarsest balancing score, but the fundamental logic behind his proof did not transcended the proof strategy for the binary case.

4.2. Balanced Mini-batch Sampling

The principles behind balancing score equips us with everything we need to develop the algorithm of balanced mini-batch sampling, which will covered below in detail.

The author proposed a two-phased method to complete the algorithm. Phase 1 requires a VAE parameterized by $\boldsymbol{\theta}$ to learn the underlying data distribution.

$$p^e_{\boldsymbol{\theta}}(X,Z|Y) = \mathbf{Pr}\left[X|Z,Y\right]p^e_{\boldsymbol{\theta}}(Z|Y) \quad (7)$$

With the VAE, the balancing score is calculated by

$$p^e(Y=y|Z) = \frac{p^e_{\boldsymbol{\theta}}(Z|Y=y)p^e(Y=y)}{\sum_{i=1}^m p^e_{\boldsymbol{\theta}}(Z|Y=i)p^e(Y=i)} \quad (8)$$

Algorithm 1 Balanced mini-batch sampling.

```
1:  $D_{\text{balanced}} = \varnothing$.
2:  for $e \in \mathcal{E}_{\text{train}}$ do
3:      Sample $k$ data points $D_{\text{sampled}}$ from $\mathcal{D}^e$.
4:      Add $D_{\text{sampled}}$ to $D_{\text{balanced}}$.
5:      for $(x^e, y^e) \in D_{\text{sampled}}$ do
6:          $y_i \sim U\{1,2,...,m\} \setminus \{y^e, y_1, ..., y_{i-1}\}$.
7:          for $i \in \{1,2,...,a\}$ do
8:              $j = \arg\min_{j\in[1,N^e]} d(b^e(z_j), b^e(z^e))$
9:              such that $y^e_j = y_i$ and $(x^e_j, y^e_j) \in \mathcal{D}^e$.
10:             Add $(x^e_j, y^e_j)$ to $D_{\text{balanced}}$.
```

Then in phase 2, the **balanced mini-batch sampling** algorithm described by Alg.1 is implemented to generate a balanced dataset.

Ideally, we will always set $a = m-1$ and find a j such that $d(b^e(z_j), b^e(z^e)) = 0$. Under such assumption, the author proved that the sampled distribution $\hat{p}^B(X,Y,Z)$ becomes the completely balanced distribution $p^B(X,Y,Z)$. We will brief on the ideas behind the proof.

Denote $D_{\text{balanced}} \sim \hat{p}^B(X,Y)$, and $\mathcal{D}^e \sim p(X,Y|e)$. Then for a perfectly matching balancing score and uniform distribution of Y given Z and E,

$$\hat{p}^B(Y|Z,e) = \hat{p}^B(Y|b^e(Z),e) = \frac{1}{m} \sim U\{1,...,m\} \quad (9)$$

It naturally follows that $\hat{p}^B(Y|Z,e) = p^B(Y|Z,e) = \frac{1}{m} = p^B(Y)$, which implies both uniform $p^B(Y)$ and conditional independence $Y \perp\!\!\!\perp_B Z$ and thus complete the proof.

4.3. Experiments

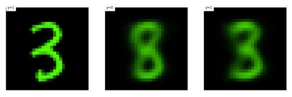
Having understood the balancing method and why such balancing is true, we explore how well balanced sampling is combined with other algorithms like ERM and IRM.

With the accuracy shown in Table.2, we find that balancing the dataset distribution is very effective for both ERM and IRM.

Table 2. Acc. Comparison

Sampling	ERM	IRM
Random	28.7	58.5
Balanced	**38.4**	**69.7**

What's more, to show the effectiveness of the trained VAE, synthetic *Colored-MNIST* images are reconstructed from the latent vector encoded by a real *Colored-MNIST* image. In Fig.(9), the leftmost image is the input, while the right two images are reconstructed from the latent vector of the input image, restricted to $Y=1$ and $Y=0$ respectively.


(a) Example 1.

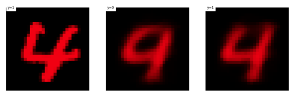

(b) Example 2.

Figure 9. Reconstruction results.

4.4. Perfectly Balanced Sampling

The idea of balanced sampling inspired us to rethink upon task 1 where context label Z is available. The major effort are paid into learning the underlying probability $\mathbf{Pr}\left[X,Y,Z\right]$, but when context label are presented, the ground truth is directly provided. Consequently, we can proceed directly to phase 2, where we do the balanced mini-batch sampling.

Moreover, it is obvious that implementing Alg.1 in *Colored-MNIST* is equivalent to sample, for every input (x,y,z), another sample (x',y',z') such that $z'=z$ and $y'=1-y$.

Notably, **this operation bears a very strong resemblance to over-sampling**, or more specifically, *random over-sampling* (ROS). So why not do ROS instead?

Def Imbalance ratio: The imbalance ratio of a class-imbalanced dataset is $\rho = \frac{\min_j |C_j|}{\max_j |C_j|}$, where C_i is the samples belonging to the i-th class.

In our problem, we have four classes, $C_{ij} = \left\{\left(x^{(k)}, y^{(k)}, z^{(k)}\right) \in \mathcal{D}^e | y^{(k)} = i, z^{(k)} = j\right\}$ for $i, j \in \{0,1\} \times \{0,1\}$. Suppose we have done ROS till $\rho = 1.0$, then it follows naturally that the distribution is balanced because $\mathbf{Pr}\left[Y = y | Z = z\right] = \mathbf{Pr}\left[Y = y\right] = 0.5$.

The technique of *random over-sampling* has been covered in detail in the github repository of one of our members, and therefore not covered in detail here. We state the conclusions without delving into discussion:

- Data augmentation is a necessary operation to prevent over-fitting when ROS is utilized.
- The most suitable way of data augmentation is random rotation for MNIST dataset.

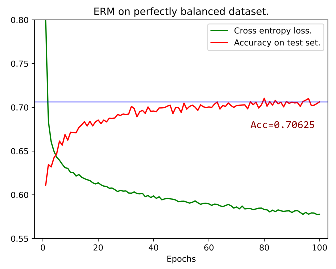


Figure 10. Result of randomly over-sampled dataset.

The experimental result of directly training a CNN on the over-sampled dataset using naive ERM shows a rather commendable accuracy of 70.6%.

4.5. Discussions

Having shown the power of causal balancing, we step further and make some discussions.

Q1: Why performance of ERM on over-sampled *Colored-MNIST* outperform that on original *Colored-MNIST* to such a significant degree?

From the experimental data, we see an accuracy of 70.6% for over-sampled dataset, while the accuracy of directly performing Alg.1 is 38.4%. Such difference implies that the proposed method fails to build an authentic balanced distribution. This demonstrates the limitation of balancing, which is not always effective enough, and thus requires the help of other algorithm-level methods.

Q2: Can balanced sampling substitute other algorithm-level methods like IRM?

No. Firstly, as is stated before, balancing is a data-level method, and thus orthogonal to other efforts paid in OoD-generalization. Secondly, in practice, there might not exist a dataset as simple as *Colored-MNIST* for the latent vector to retrieve key features and allowing for effective reconstruction. Therefore, the proposed method is better used as a supplement for algorithm-level methods.

5. Conclusion

In the report, we have delved deeply into comprehending the *Colored-MNIST* and the *Out-of-Distribution generalization* problem it presented. Based on two different assumptions (the context label being either available or unavailable), we have developed two methods to achieve decent accuracies on the test set, one attempting at locating causal feature from algorithm level, while the other tries to modify the training dataset. During the process of exploring solutions and conducting literature survey, we have deepened our understanding of causal inference, spurious correlations and OoD generalization.

References

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2020.

[2] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization, 2020.

[3] Xinyi Wang, Michael Saxon, Jiachen Li, Hongyang Zhang, Kun Zhang, and William Yang Wang. Causal balancing for domain generalization, 2023.