CSC494-spatialpfn

Richard Guo

September 2025

Contents

1 Introduction and Setup

The goal of this project is to extend CausalPFN to spatial/temporal data. We will be using resources such as Meta's DINOv3, TabiCL, ComputeCanada, and more.

1.1 Compute Canada Setup & Must-Knows

1.1.1 Virtual Environment

Create a virtual environment in Compute Canada, and activate it with

```
source ~/envs/env_name/bin/activate
```

Then, use `pip` to install any necessary libraries. Make sure to activate the environment in the `.sh` file

1.1.2 Data & Models Import

All connections to the internet have to be established **before** queuing jobs on Compute Canada. For example, instead of

`from TabiCL import TabiCLClassifier`,

the HuggingFace link to TabiCL must first be downloaded to the cache in ComputeCanada and the `HF_HOME` variable must be set in the `.sh` file

```
python -c "from huggingface_hub import snapshot_download;
snapshot_download(repo_id='jingang/TabICL-clf',
cache_dir='/home/username/huggingface_cache')"
```

```
export HF_HOME=/home/richguo/huggingface_cache
```

Data like MNIST also needs to be downloaded in the shell before queuing.

```
python -c "from keras.datasets import mnist; mnist.load_data()"
```

1.1.3 DINOv3

In order to use DINOv3 in any script on Compute Canada, the local model must first be downloaded and uploaded to some folder in Compute Canada with

```
scp -r path_to_local_dinov3 username@cluster.alliancecan.ca:~
/projects/def_rahulgk/username.
```

Model weights (.pth files) must also be downloaded.

From there, the `.py` script can load the DINOv3 model by pointing to the model cache and weights.

1.1.4 Setting TabiCL to CUDA

Check if Compute Canada's wheel-built version of PyTorch supports CUDA; if not, install a different version of PyTorch

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

1.1.5 Job Parameters

```
 #!/bin/bash
# SBATCH --job-name=spatialpfn
# SBATCH --account=def-rahulgk # your PI's account
# SBATCH --time=02:00:00 # hh:mm:ss, adjust as needed
# SBATCH --cpus-per-task=8 # number of CPU cores # SBATCH --mem=64G
# SBATCH --gres=gpu:1 # request GPU if needed; omit if CPU-only
# SBATCH --output=spatialpfn_%j.out # standard output file
```

1.1.6 Queuing & Monitoring Jobs

Jobs can be queued with

```
sbatch run_spatial-pfn.sh
```

To monitor jobs,

```
squeue -u $USER
```

or

```
squeue --start -u $USER
```

to see expected start times.

1.1.7 Debugging in CC

Make sure to add the `flush=True` parameter to all `print` statements in the .py file, otherwise nothing will be printed until the program finishes running (if program fails, nothing will be printed at all)

2 Regular MNIST

2.1 MNIST into TabiCL

Simple pipeline taking the MNIST dataset, testing different PCA levels, then feeding it through TabiCL to evaluate accuracy. Note: use ComputeCanada to test higher sample sizes (ex. n > 5000)

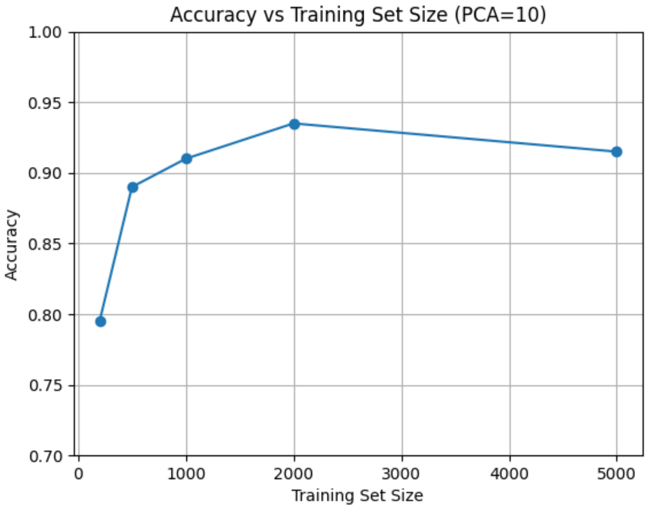


Figure 1: Training size vs TabiCL accuracy for 10 principal components of MNIST.

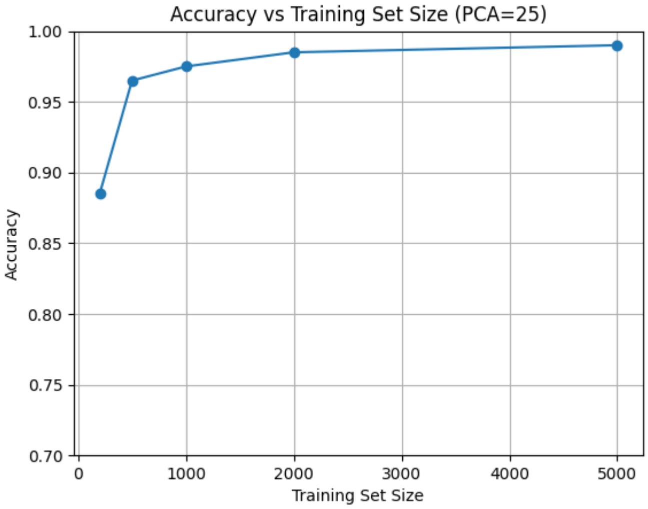


Figure 2: Training size vs TabiCL accuracy for 25 principal components of MNIST.

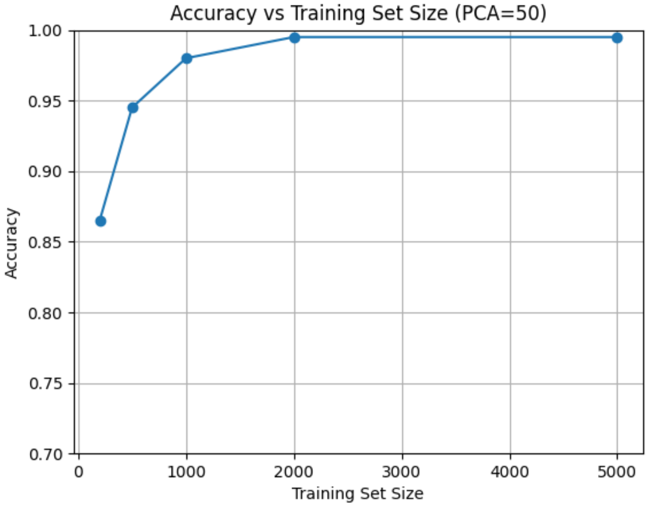


Figure 3: Training size vs TabiCL accuracy for 50 principal components of MNIST.

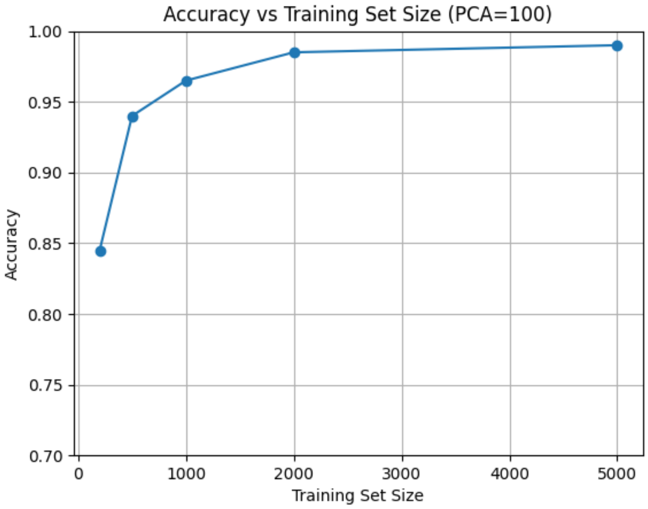


Figure 4: Training size vs TabiCL accuracy for 100 principal components of MNIST.

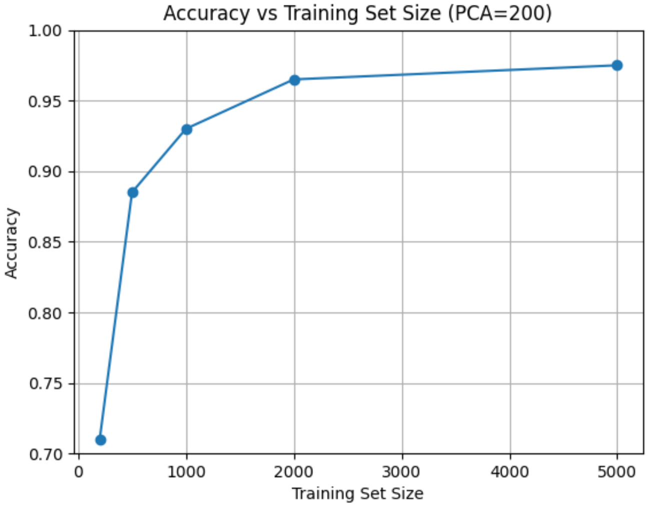


Figure 5: Training size vs TabiCL accuracy for 200 principal components of MNIST.

2.2 MNIST into DINOv3 into TabiCL

Taking MNIST dataset and feeding it into DINOv3 (unsupervised vision transformer), then taking DINOv3 vector embeddings and feeding it through TabiCL to evaluate accuracy. Note: PCA may be used on the vector embeddings

2.2.1 Accuracy scores for DINOv3 embeddings

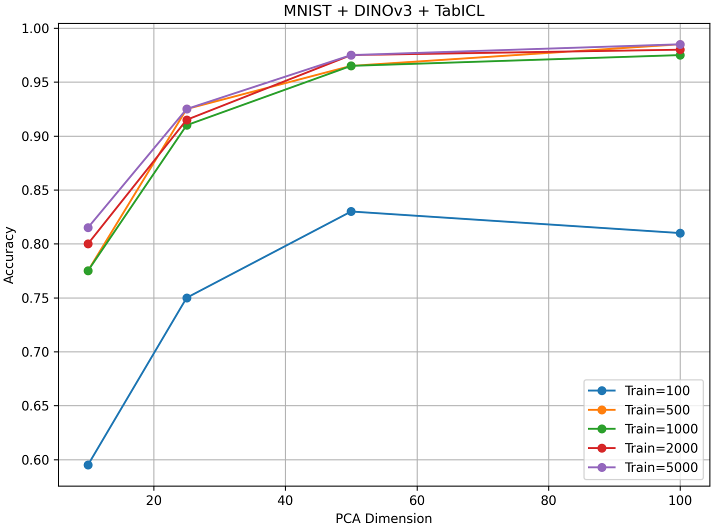


Figure 6: Training size vs TabiCL accuracy on DINOv3 processed MNIST with various PCA dimensions.

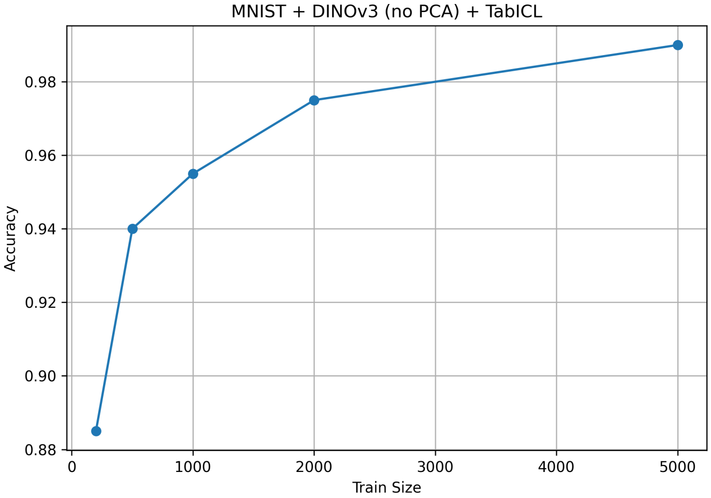
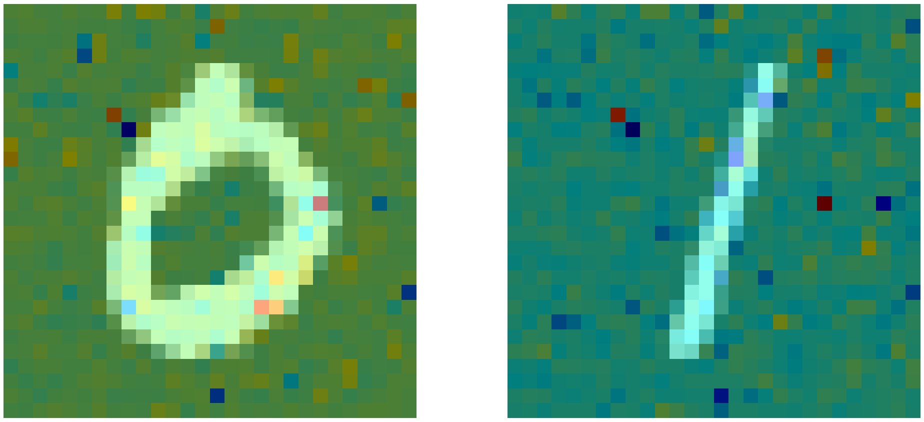


Figure 7: Training size vs TabiCL accuracy on DINOv3 processed MNIST. Accuracy at 0.99 for sample size of 5000 - Worth looking into even higher sample sizes to find plateau

2.2.2 Rollout-Attention Maps for MNIST Digits

Attention maps are averaged out across all layers, so some of the digits may look slightly different depending on how they are drawn.

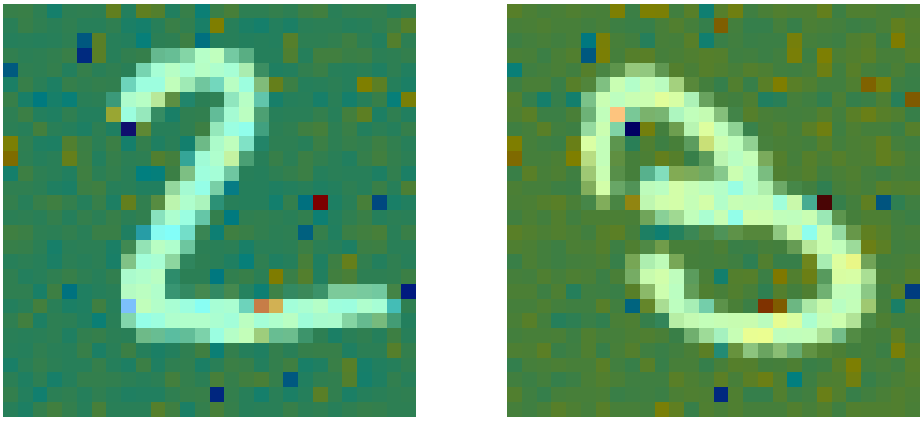
Darker colours: less attention, brighter colours: more attention



(a) Digit 0

(b) Digit 1

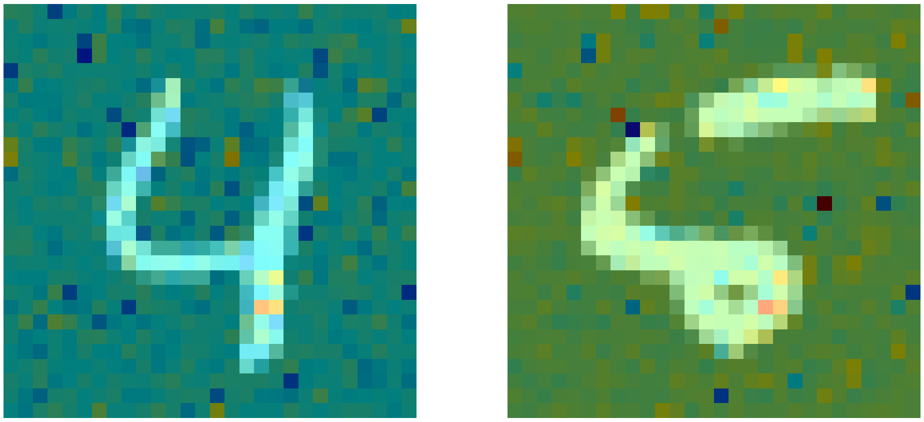
Figure 8: DINOv3 attention-rollout maps for digits 0 and 1.



(a) Digit 2

(b) Digit 3

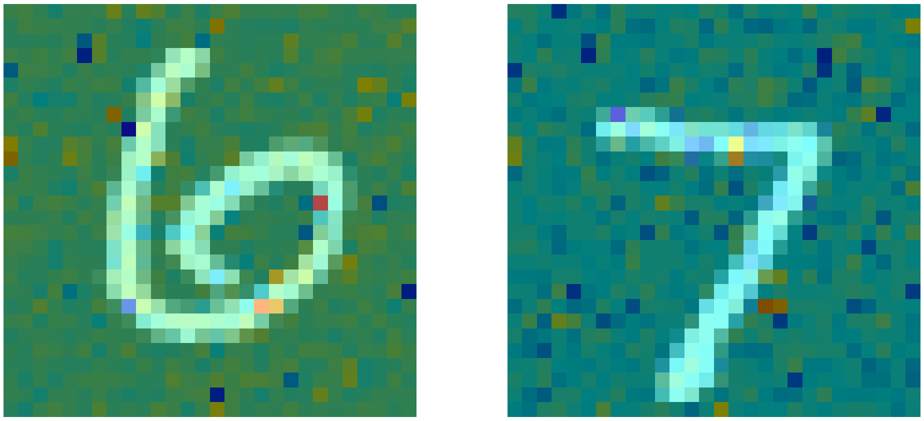
Figure 9: DINOv3 attention-rollout maps for digits 2 and 3.



(a) Digit 4

(b) Digit 5

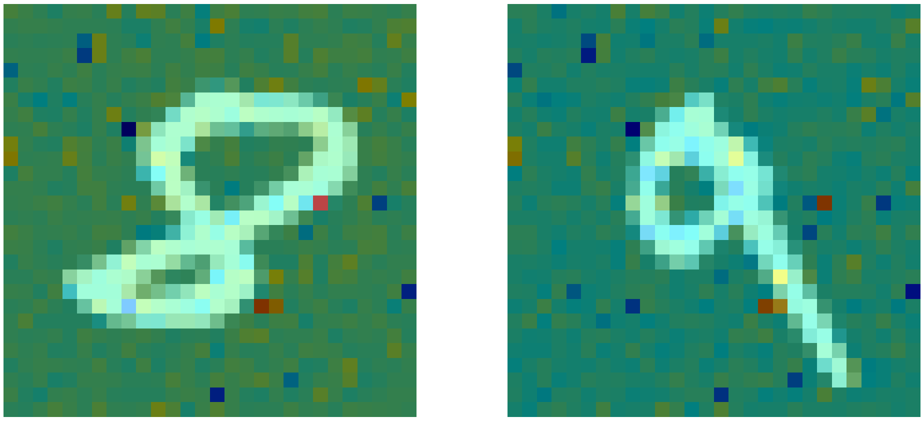
Figure 10: DINOv3 attention-rollout maps for digits 4 and 5.



(a) Digit 6

(b) Digit 7

Figure 11: DINOv3 attention-rollout maps for digits 6 and 7.



(a) Digit 8

(b) Digit 9

Figure 12: DINOv3 attention-rollout maps for digits 8 and 9.

3 MNIST with Synthetic Treatment Effects

*Note: left for CSC495

3.1 Big Picture

We want a synthetic causal dataset where:

- Treatment is encoded spatially as an X mark,
- Outcome is encoded spatially as an O mark,

- There's also a confounder that affects both treatment and outcome,
- The whole thing is visible in an MNIST-like image so we can embed it with DINOv3 and later analyze causal structure with CausalPFN.

The design has to satisfy both causal inference needs (non-randomized treatment, confounding, heterogeneous effects) and vision needs (clear markers that the model can detect).

3.2 Mapping Class Labels to X's and O's

We map each digit class (0–9) to a pair of coordinates.

This ensures each digit gets a different spatial pattern. If all digits have the same X and O positions, there would be no class-specific variation for DINOv3 to learn. By tying marker positions to digit identity, we create a model where the "underlying group" (digit class) affects both the geometry and the causal process.

If we just placed digit 0 at the top-left, 1 at top-middle, etc., DINOv3 could memorize a simple grid pattern. Using sine/cosine coefficients which are not completely class-reliant creates indicators in a more random way. This avoids overlaps and makes it harder for the model to "cheat".

$< u, v >_X = < L + sin(L), L + cos(L) >$

$< u, v >_O = < L + cos(L), L + sin(L) >$

Where L represents the class label (0-9). Coefficients may be introduced to fine-tune placements in the future.

3.3 The Confounder $\mathtt{z}$

We can compute $\mathtt{z}$ using properties that the images already have:

- Mean intensity (brigher handwriting = higher $\mathtt{z}$)
- Vertical center (higher vs lower writing placement)

By basing $\mathtt{z}$ on the actual image, we guarantee it is encoded in the visual appearance, so DINOv3 embeddings have access to it. This way, if you ignore $\mathtt{z}$, you get biased causal estimates.

$\mathtt{z} = \sigma(\psi_1 \cdot \text{mean} + \psi_2 \cdot \text{vcenter}) \in (0, 1)$

3.4 Treament $\texttt{T}$

We define

$p(\texttt{T} = 1) = \sigma(\alpha_0 + \alpha_1 \texttt{z} + \alpha_2 x_X + \alpha_3 x_Y)$

where α_0 represents the bias, α_1 makes treatment probability increase with confounder $\texttt{z}$, and α_2 and α_3 introduce some spatial bias; treatments are more likely at some positions than others.

The X marker will be thick if $\texttt{T} = 1$ and thin otherwise.

3.5 Outcome $\texttt{Y}$

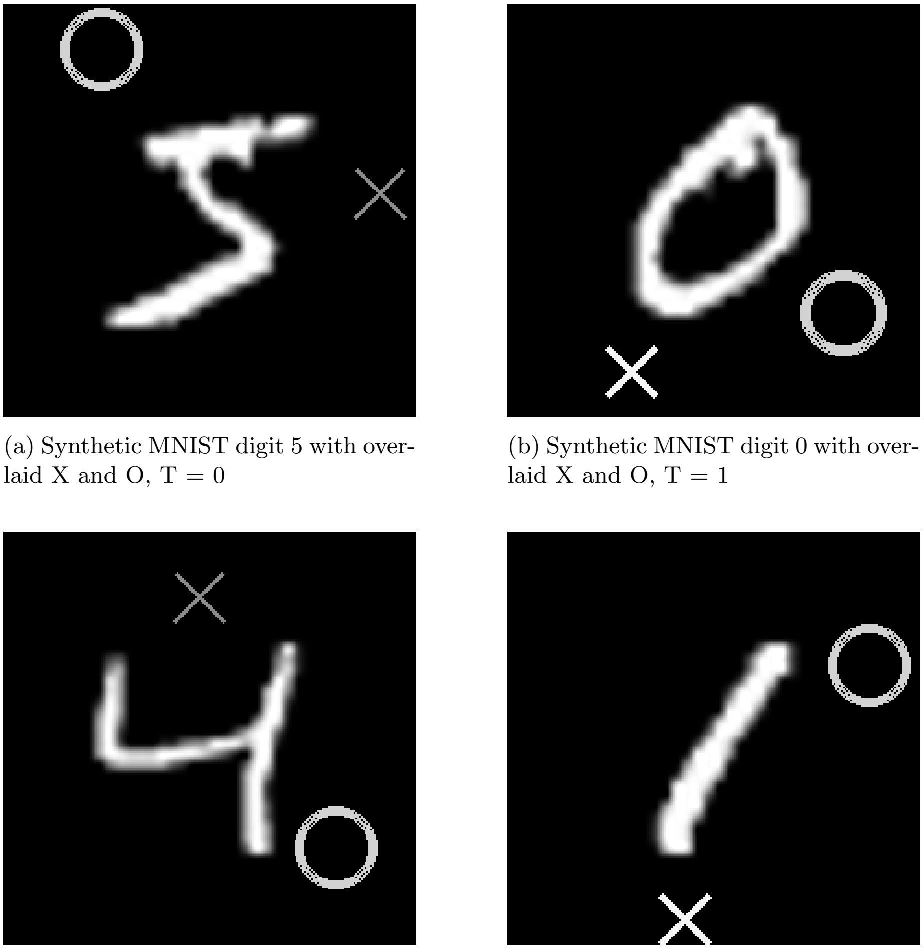
We define

$\texttt{Y} = \beta_0 + \beta_1 \texttt{T} + \beta_2 \texttt{z} - \beta_3 \text{dist}(X, O) + \epsilon$

Where $\beta_1 > 0$ ensures that treated samples have higher outcomes, $\beta_2 > 0$ creates confounding bias, and $\beta_3 > 0$ forces embeddings to encode geometry, not just the existence of X's and O's.

The thickness of the drawn O will scale with the value that $\texttt{Y}$ takes, representing how the outcome changes with respect to the treatment.

3.6 Synthetic Dataset Examples



(a) Synthetic MNIST digit 5 with overlaid X and O, T = 0

(b) Synthetic MNIST digit 0 with overlaid X and O, T = 1

(c) Synthetic MNIST digit 4 with overlaid X and O, T = 0

(d) Synthetic MNIST digit 1 with overlaid X and O, T = 1

Figure 13: Examples from the synthetic MNIST causal dataset with X (treatment) and O (outcome) markers.

4 Simple Causal MNIST Dataset & CausalPFN

4.1 Causal Structure

Using the regular MNIST dataset, we aim to define T and Y using both X and D, where $d \in D$ is the class label (in this case, the number in the MNIST image).

For each digit d, we draw weights and biases (w_d, b_d) and use them to create

$$\ell_i = w_{D_i}^\top x_i + b_{D_i},$$

where x_i is the i-th image. Within each digit class, we standardize the logits l_i,

$$z_i = \frac{\ell_i - \mu_{D_i}}{\sigma_{D_i} + \varepsilon},$$

and then solve for a per-class shift s_{D_i} so that the marginal treatment probability ragnes from 0.25 to 0.75):

$$E\big[\sigma(z_i + s_{D_i}) \,\big|\, D_i = d\big] \approx p_d,$$

where $\sigma(\cdot)$ is the logistic sigmoid. The construction of p_d is a little complex, but is simply used so that T doesn't have too extreme class imbalance. Finally, treatment is sampled as

$$T_i \,\big|\, x_i, d \sim \text{Bernoulli}\big(\sigma(z_i + s_d)\big).$$

This construction makes T depend on both D and X, meaning different digits have different baseline treatment rates, and even within a digit, the specific properties and randomness affects the probability of treatment.

For Y, we experiment with multiple different constructions; however each construction must have Y depend on both D and T.

4.1.1 Simple additive Y outcome

We model `Y` using a simple additive model

$\mathtt{Y} = \alpha T + f(D) + \epsilon$

where we set $\alpha = 2$ and $f(D)$ is a simple MLP to introduce non-linearity. In this model, the treatment T and the confounding D are separate.

Digit	n	ATE_hat	CATE_mean	CATE_std
0	2018	1.9843	1.9920	0.00165
1	2229	2.0075	1.9961	0.00339
2	1943	2.0069	2.0160	0.00194
3	2020	2.0033	2.0086	0.00222
4	1957	1.9983	2.0013	0.00222
5	1862	1.9727	1.9700	0.00314
6	1995	2.0007	1.9943	0.00161
7	2022	1.9979	2.0000	0.00221
8	1967	1.9872	1.9822	0.00160
9	1987	2.0054	2.0103	0.00154
999	20000	2.0092	2.0077	0.14292

Table 1: Estimated ATE and CATE statistics by digit.

From this table, we can see that CausalPFN successfuly recovers the treatment effect ($\alpha = 2$) in both the estimated ATE and the estimated CATE.

4.1.2 Heterogeneous Y

We model `Y` with

$\texttt{Y} = f(T, D) + \epsilon$

$$f(T, D) = h(D) + g(D) \cdot T + \sin\big(\pi T + 0.3D\big)$$

where

$h(D) = \mathrm{MLP}_{\mathrm{base}}(D)$
$g(D) = \mathrm{MLP}_{\mathrm{treat}}(D)$

where $h(D)$ learns a baseline outcome level for each digit, so different classes naturally start at different values even before treatment is applied. The second network $g(D)$ controls how strongly each digit responds to treatment, which lets the model assign larger or smaller treatment effects depending on the digit identity. In this model, the treatment and outcome have interaction terms, making it harder to recover the true effect.

Digit	n	ATE_hat	CATE_mean	CATE_std	True_effect
0	2018	0.1295	0.1371	0.00175	0.1288
1	2229	-0.4077	-0.4175	0.00351	-0.4147
2	1943	-0.9619	-0.9528	0.00194	-0.9595
3	2020	-1.3238	-1.3185	0.00222	-1.3237
4	1957	-1.6494	-1.6464	0.00222	-1.6495
5	1862	-1.8020	-1.8046	0.00314	-1.7996
6	1995	-1.7641	-1.7705	0.00161	-1.7646
7	2022	-1.5677	-1.5658	0.00221	-1.5678
8	1967	-1.1985	-1.2034	0.00160	-1.1983
9	1987	-0.7660	-0.7611	0.00154	-0.7672
999	20000	-1.1127	-1.1149	0.56721	–

Table 2: Comparison of estimated ATE/CATE estimates and true effects by digit.

Table 2 shows that, even with a more complex Y model, CausalPFN still successfully recovers the true effect with near-perfect accuracy.

4.1.3 Varying Degrees of Confounding

We model `Y` with

$\texttt{Y} = f(T, D) + \alpha \cdot C(D) + \epsilon$

where $f(T, D)$ is defined by the same two MLPs and sin function from `4.1.2` and $C(D)$ is a confounding variable modeled as $C(D) = tanh(D \cdot w_c)$ where w_c is randomly sampled from a $N(0, 1)$ distribution. We control the confounding level with α.

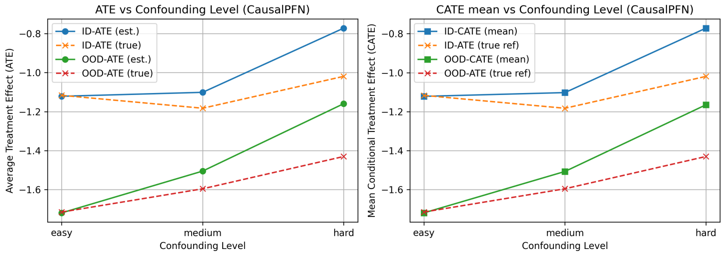


Figure 14: ID and OOD estimations vs ground truth, sorted by confounding level (hardness)

From Figure 14, we can see that CausalPFN recovers the causal effect as expected; the higher the confounding level, the more it struggles to pull the ground truth. Forr easy and medium confounding, CausalPFN is pretty accurate with its estimations, while the estimation becomes much worse for hard confounding.

5 List of Causal Datasets

Dataset	Link	Free	T (Treatment)	Y (Outcome)	Confounders	Notes / Description
Causal Chamber	`https://github.com/juangamella/causal-chamber`	Yes	Light-tunnel actuator (polarizer angle, color)	Image-derived intensity / task label	Other actuators, ambient factors	Real physical light-tunnel system with known interventions and ground-truth graphs.
Causal3DIdent	`https://proceedings.neurips.cc/paper/2021/file/8929c70f8d710e412d38da624b21c3c8-Paper.pdf`	Yes	Intervened latent (hue/pose/light)	Class/regression from rendered image	Lighting/background	Synthetic 3D dataset with explicit causal latents and interventions.
Causal3D	`https://zenodo.org/records/4566200`	Yes	Light position	Shadow length / intensity	Object shape, surface, background	Simple 3D scenes designed to study causal effects of illumination on shadows.
CausalCity	`https://causalcity.github.io/dataset.html`	Yes	Scenario/weather/time-of-day	Risk/scene property or proxy label	Road layout, object mix, viewpoint	Driving simulator dataset with controllable causal interventions.
Causal Circuit	`https://github.com/szabgab/causal-circuit`	Yes	Component value change / voltage input	Output voltage / current	Circuit topology, temperature	Synthetic electrical circuit simulator with known causal graphs.
CLEVR	`https://cs.stanford.edu/people/jcjohns/clevr/`	Yes	Object attribute (color, size, position)	Scene question answer / label	Lighting, scene complexity	Synthetic visual reasoning dataset useful for studying visual causal reasoning.
Waterbirds	`https://github.com/deeplearning-wisc/Spurious_OOD`	Yes	Background (water vs. land)	Bird class (waterbird/landbird)	Location, lighting, viewpoint	Combines CUB and Places datasets to test spurious correlations.
CelebA	`https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`	Yes†	Visual attribute (e.g., hair color, glasses)	Target attribute / identity proxy	Pose, age, gender, lighting	Large-scale facial attributes dataset with strong attribute correlations.
Camelyon17 (WILDS)	`https://wilds.stanford.edu/datasets/#camelyon17`	Yes	Hospital center / scanner / stain	Metastasis label	Patient mix, slide preparation	Histopathology tiles with domain shifts across hospitals.
iWildCam (WILDS)	`https://lila.science/datasets/iwildcam-2022/`	Yes	Habitat/season/location	Species label/presence	Camera site, time-of-day, motion	Camera-trap dataset for species classification under domain shifts.
PovertyMap (WILDS)	`https://proceedings.mlr.press/v139/koh21a/koh21a-supp.pdf`	Yes	Country (policy/region) or urban/rural	Asset wealth index	Geography, season, clouds	Satellite imagery linked with socioeconomic survey data.
NICO / NICO++	`https://github.com/xxgege/NICO-plus`	Yes	Context (scene/background)	Object class	Scene style, weather, device	Non-IID dataset designed to study context bias and OOD generalization.
Colored MNIST	`https://colab.research.google.com/github/reiinakano/invariant-risk-minimization/blob/master/invariant_risk_minimization_colored_mnist.ipynb`	Yes	Digit color	Digit label	Domain/color correlation	Synthetic digit dataset for causal/spurious correlation testing.
dSprites / 3D-Shapes / MPI3D	`https://github.com/google-deepmind/dsprites-dataset`	Yes	One latent factor (e.g., rotation, scale, hue)	Shape/pose/other latent	Remaining latent variables	Fully disentangled generative datasets with controlled factors.
MIMIC-CXR	`https://physionet.org/content/mimic-cxr/2.0.0/`	Yes (registration)	View position (AP vs. PA)	Pathology label	Age, gender, hospital device	Large medical imaging dataset for chest X-rays; confounding due to acquisition view.
iNaturalist	`https://github.com/visipedia/inat_comp`	Yes	Month or location	Species class	Photographer bias, lighting, device	Real-world species classification dataset with strong geographic/temporal confounding.

Table 3: Image(-like) datasets commonly used for causal or spurious correlation studies, with suggested causal casts for treatment (T), outcome (Y), confounders, and brief dataset descriptions. †CelebA is free for research; registration may be required.

6 Causal Chamber

6.1 Background & Methodology

The Causal Chamber dataset is a real, physical light–tunnel system designed to serve as a benchmark for causal reasoning in the wild. The Causal Chamber presents a controllable optical system where the causal structure is created using physics.

The system consists of a rotatable polarizer, configurable LED light sources, and a mounted camera that records the resulting light patterns. Each configuration (e.g., polarizer angle, LED color or brightness) has a causal influence on the measured sensor outputs like pixel intensities. Their causal effects are governed by physics; for example, polarization intensity behaves according to Malus' Law, allowing us to calculate a ground truth.

6.1.1 Causal Structure

In our work, we focus on the `lt_camera_v1` part of the dataset, which consists of camera images and their corresponding chamber actuator settings. We treat the polarizer angle θ treatment and define a binary treatment variable

$$T = \mathbf{1}\{\theta > \theta_0\},$$

where θ_0 is a fixed threshold. While this setup is not perfect as it allows for similar angles to have different T, the hope is that CausalPFN is able to recover the general pattern of Malus' Law.

The outcome Y is taken to be image-derived brightness, computed from the camera frame and normalized to account for exposure and ambient conditions. Although Y is scalar, its dependence on θ is nonlinear, and the mapping from the raw image to brightness is intentionally left as part of the learning problem. Because the camera image encodes not only θ but also factors such as LED characteristics and environmental noise, the model must implicitly learn to disentangle these influences.

6.1.2 Ground Truth

The underlying physics of the setup follows Malus' Law. The ground-truth causal effect depends on the angle between the two polarizers,

$$\tau_{\text{true}}^{\text{raw}} = \cos^2(\theta_2 - \theta_1),$$

where θ_1 = `pol_1` and θ_2 = `pol_2`. A standardized version is used for CATE evaluation.

6.2 Experiment 1: Treatment Based on Angle Difference

In this experiment, we define `T` with the angle between the polarizers,

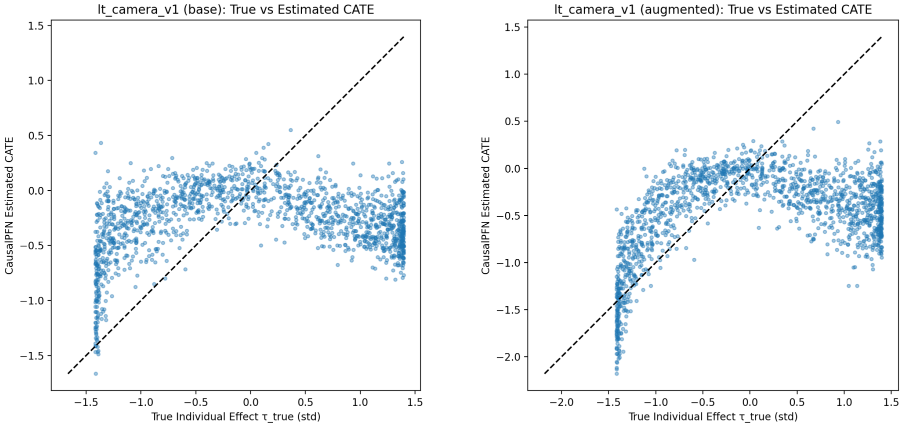
$$\Delta\theta = \theta_2 - \theta_1.$$

and take the treatment to be

$$T = 1\{\Delta\theta > \text{median}(\Delta\theta)\},$$

which better captures high- vs. low-angle differences. We have also augmented the embeddings with the `pol_1` and `pol_2` vectors to see if ”cheating” makes a difference.

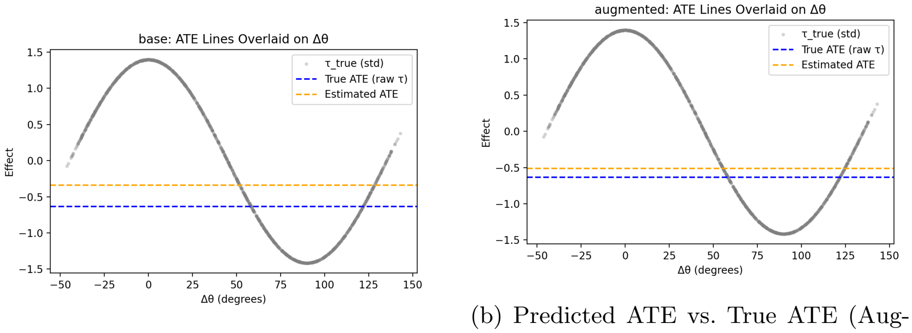
6.2.1 Experiment 1 Visualizations



(a) Predicted CATE vs. True Effect (Base)

(b) Predicted CATE vs. True Effect (Augmented)

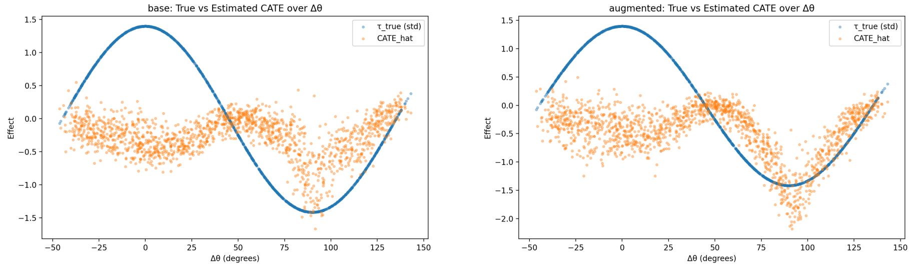
Figure 15: CATE scatterplots comparing base and augmented versions.



(a) Predicted ATE vs. True ATE (Base)

(b) Predicted ATE vs. True ATE (Augmented)

Figure 16: ATE comparison plots for base and augmented datasets.



(a) Predicted CATE vs. Standardized True Effect (Base)

(b) Predicted CATE vs. Standardized True Effect (Augmented)

Figure 17: Comparison of predicted CATE curves against standardized true effects.

6.2.2 Experiment 1 Results

The scatter in Figure 15.a shows that the predicted CATE mostly fails to capture the true effect, especially at higher values of τ, though in the augmented version, the scatter does somewhat capture the true effect for low values of τ, but still fails for higher values.

The ATE results show that the estimated ATE is quite close to the true ATE, and the augmented version shows an even closer estimation.

While the predicted CATE fails to fully capture the causal effect of $\Delta\theta$ on the observed brightness, the effect at $\Delta\theta > 50$ is somewhat captured. This is made more prominent in the augmented version, where the estimated CATE scatter more closely follows the sinusoidal true effect dictated by Malus' Law.

6.3 Experiment 2: K-Bucket Median-Split Treatments

The fourth experiment aims to study how treatment effects vary across the full range of relative polarizer angles. Instead of defining a single global treatment, we partition the data into K buckets along the angle-difference axis,

$$\Delta\theta = \theta_2 - \theta_1,$$

where each bucket captures a different section of the wave-like Malus' Law.

6.3.1 Experiment 2 Visualizations

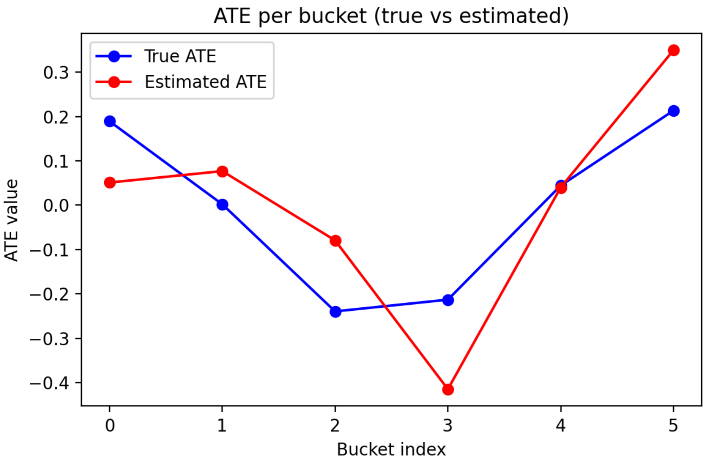


Figure 18: Estimated ATE vs. true ATE across all 5 buckets.

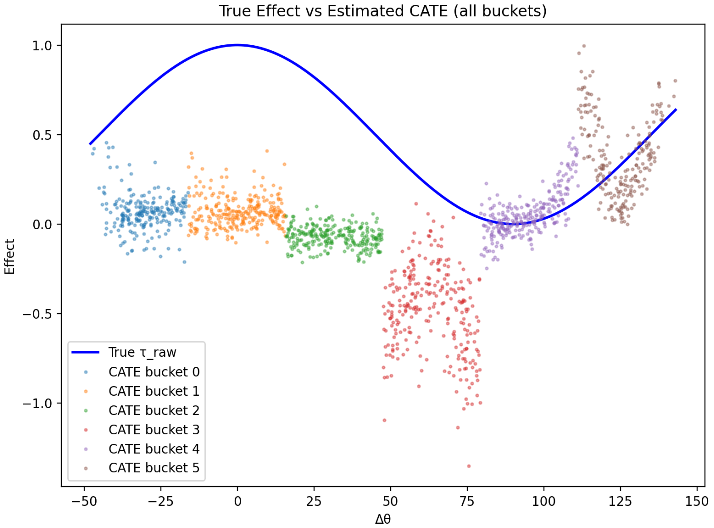


Figure 19: CATE estimates vs. True Effect across all 5 Buckets.

6.3.2 Experiment 2 Results

While the estimated ATE closely follows the true ATE in Figure 18, the CATE estimates fail to capture the sinusoidal curve of the true effect in Figure 19.

7 Next Steps (CSC495)

Benchmarking CausalPFN across multiple new datasets

Exploring our original goal of extracting X, T, and Y representations from unsupervised images using DINOv3

Any other tasks/ideas that may come up during the term

8 References

facebookresearch. (2025). *DINOv3: Reference PyTorch implementation and models for DINOv3.* GitHub. `https://github.com/facebookresearch/dinov3`

Siméoni, O., Vo, H. V., Seitzer, M., Baldassarre, F., Oquab, M., Jose, C., Khalidov, V., Szafraniec, M., Yi, S., Ramamonjisoa, M., Massa, F., Haziza, D., Wehrstedt, L., Wang, J., Darcet, T., Moutakanni, T., Sentana, L., Roberts, C., Vedaldi, A., Tolan, J., Brandt, J., Couprie, C., Mairal, J., Jégou, H., Labatut, P., & Bojanowski, P. (2025). *DINOv3*. arXiv preprint arXiv:2508.10104. https://arxiv.org/abs/2508.10104

Pearl, J. (2010). An introduction to causal inference. *International Journal of Biostatistics, 6*(2), Article 7. https://pmc.ncbi.nlm.nih.gov/articles/PMC2836213/

Facure, M. (2025). *Introduction to causality.* In *Python causality handbook.* https://matheusfacure.github.io/python-causality-handbook/01-Introduction-To-Causali html

Callis, S. J. (2025). *Attention for vision transformers, explained.* Medium. https://medium.com/data-science/attention-for-vision-transformers-explained-70f83984