Finding Real Uncertainties From Lensing Simulations

Towards Real Data UQ With Domain-Adaptive Neural Nets

Shrihan Agarwal

Alex Ćiprijanović

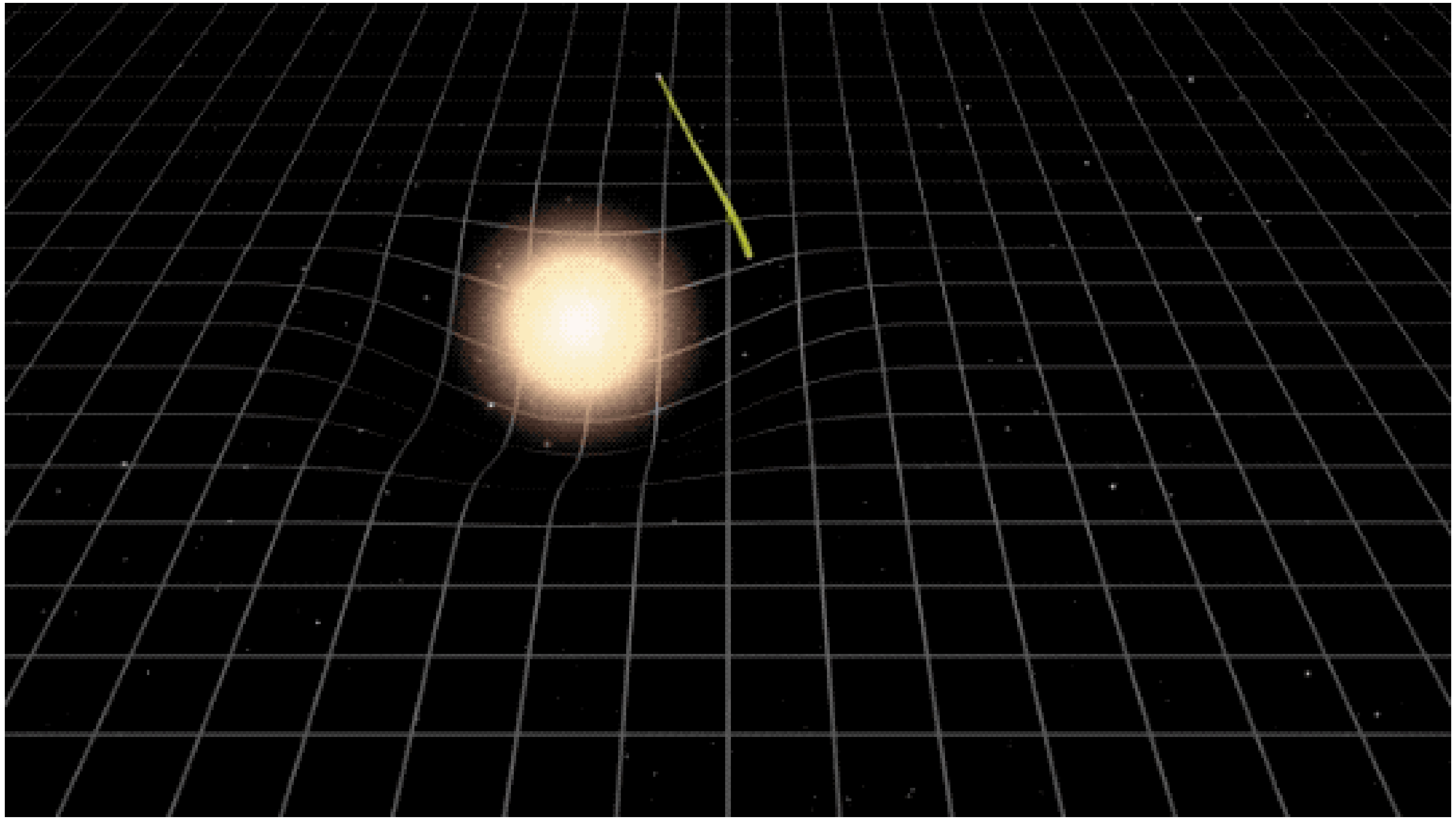
Brian Nord

The University of Chicago

Fermilab

Strong Lensing: Galaxies Bend Light, Creating Arcs

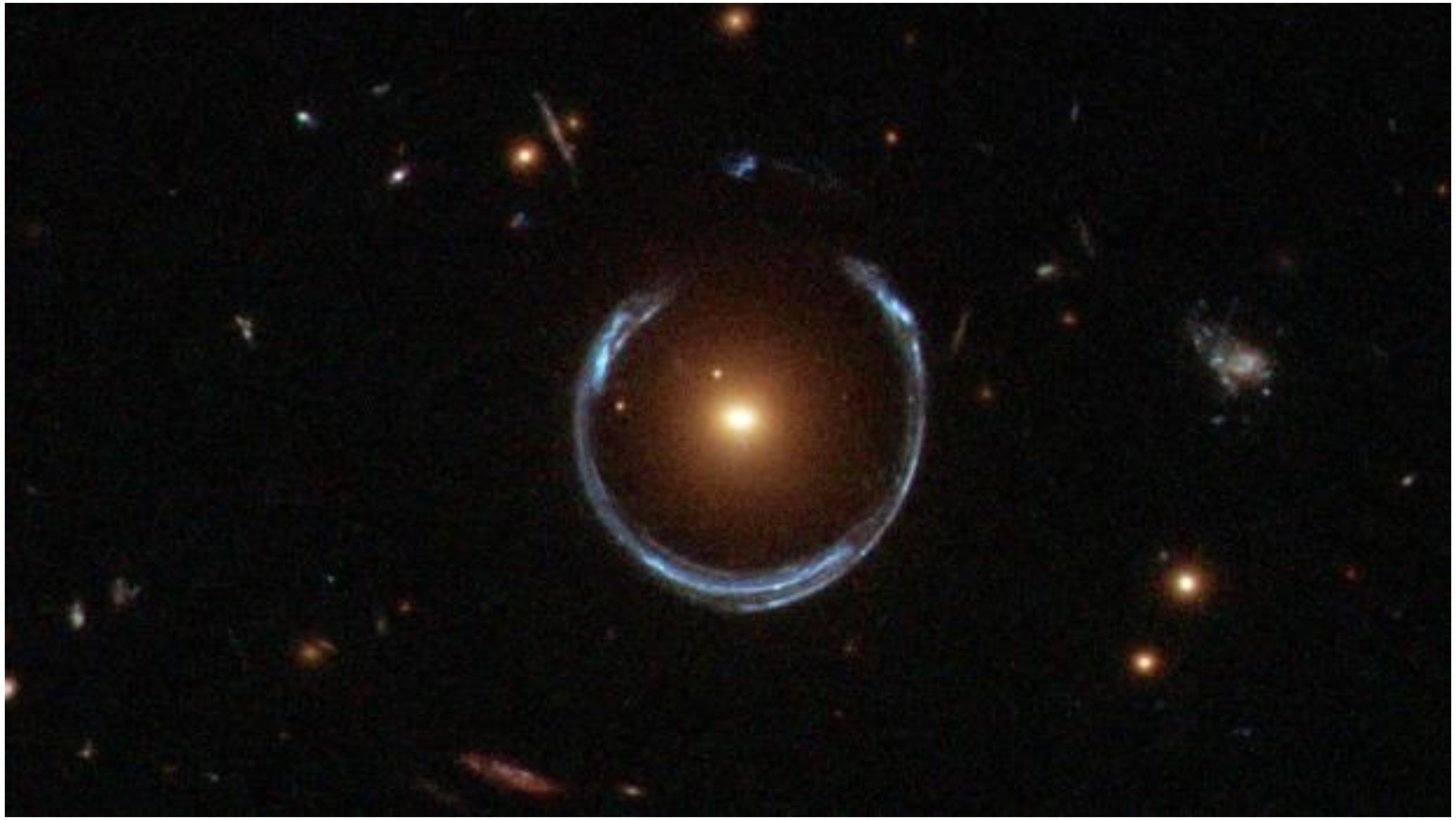
- A galaxy can bend light from one right behind it
- The background galaxy light is sheared
- Making arcs around the foreground galaxy



Credit: NASA, ESA, and Goddard Space Flight Center/K. Jackson

Strong Lensing: Galaxies Bend Light, Creating Arcs

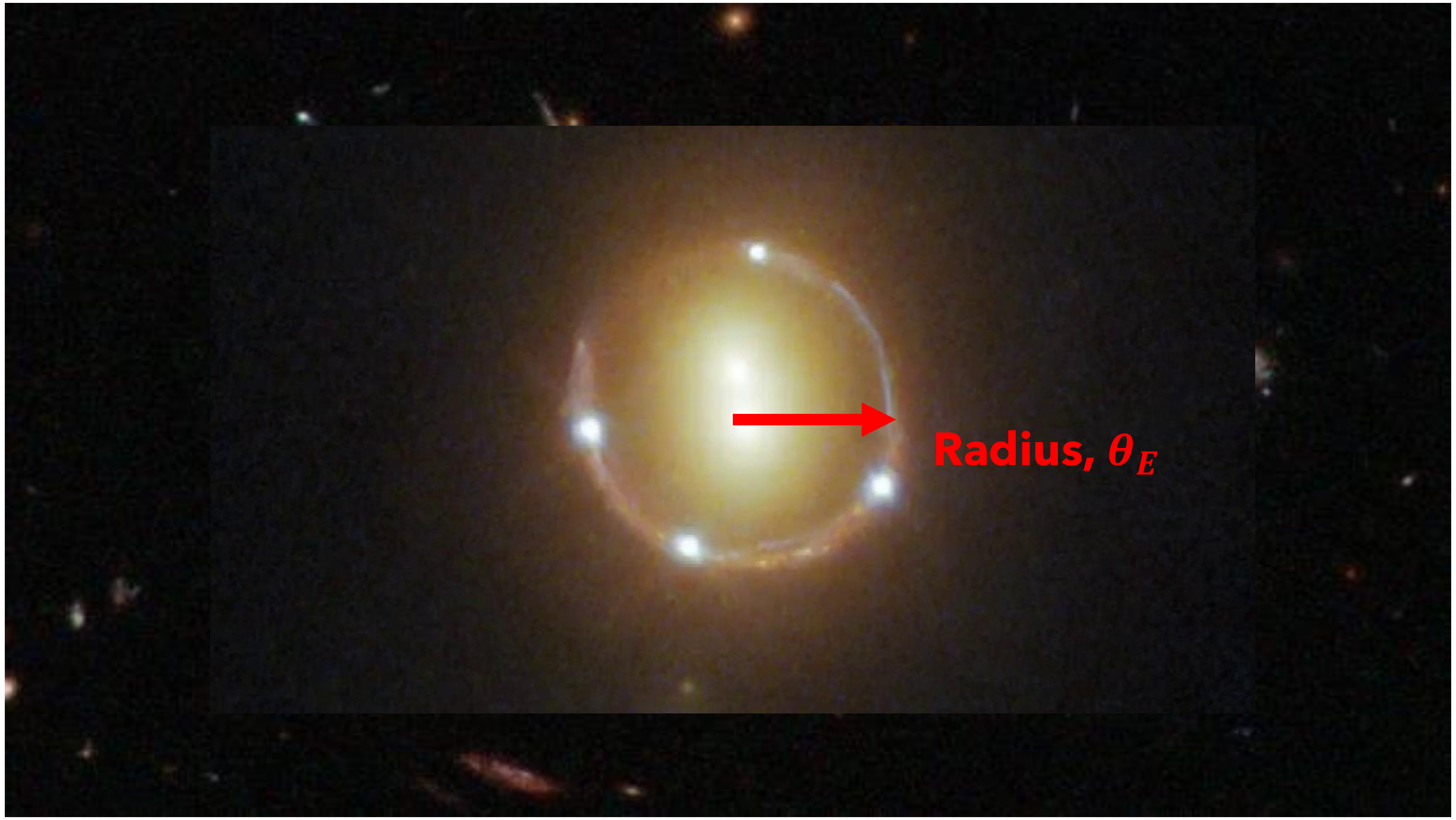
- A galaxy can bend light from one right behind it
- The background galaxy light is sheared
- Making arcs around the foreground galaxy



Credit: NASA, ESA, and Hubble Space Telescope

Strong Lensing: Galaxies Bend Light, Creating Arcs

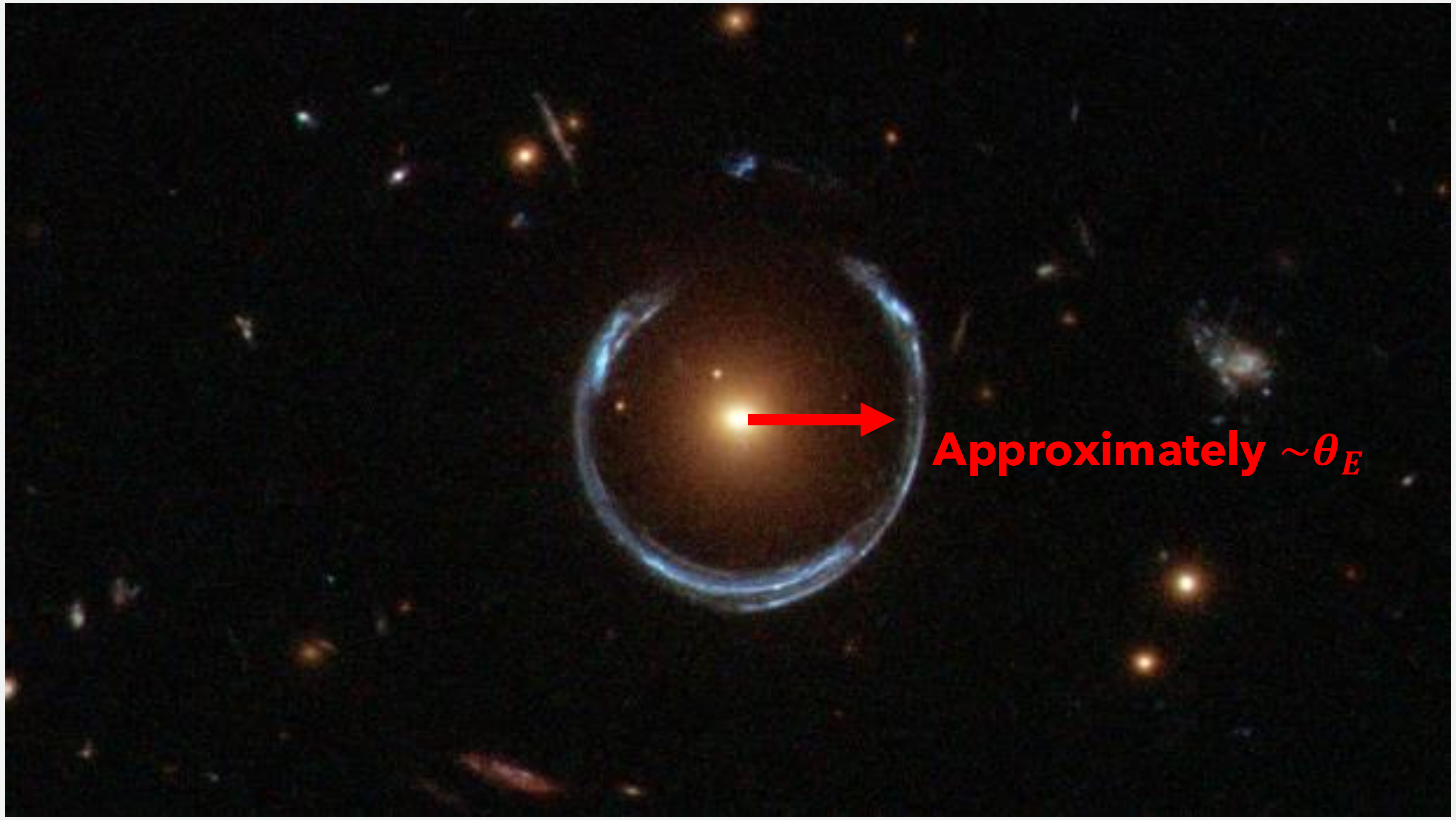
- A galaxy can bend light from one right behind it
- The background galaxy light is sheared
- Making arcs around the foreground galaxy



Credit: NASA, ESA, and Hubble Space Telescope

Strong Lensing: Galaxies Bend Light, Creating Arcs

- A galaxy can bend light from one right behind it
- The background galaxy light is sheared
- Making arcs around the foreground galaxy



Credit: NASA, ESA, and Hubble Space Telescope

New Telescopes Will Find Over 10^5 Similar Systems

Modeling these lenses can:

Test the expansion rate of the universe, by measuring distances

Offer a magnified (lensed) view of small, far-away galaxies

Study how dark matter is structured around galaxies

The Challenge of Uncertainties: Simulation-Trained NNs

Methods to Get Uncertainties on theta_E

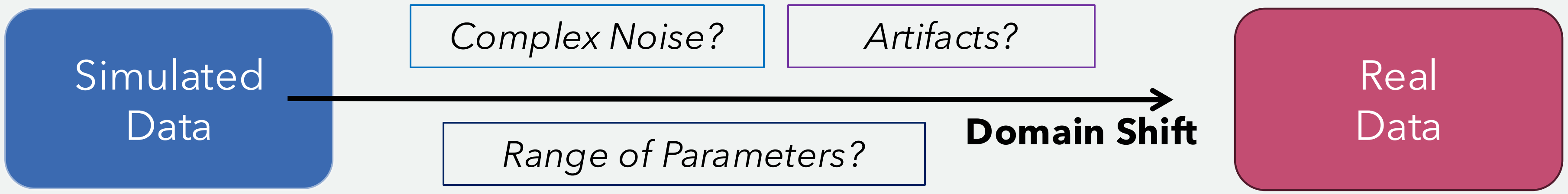
- MC Dropout
- Bayesian Neural Networks
- Deep Ensembles

...

- **Mean and Variance Estimation (MVE)**

Simulation-Trained NNs

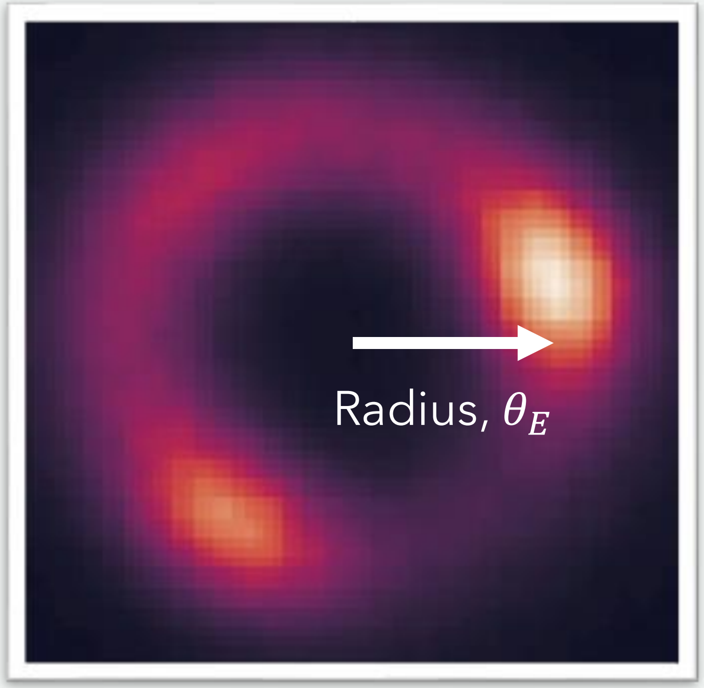
- Need to make simulations tuned carefully to the real data
- Real data may have quirks that are hard to simulate
- This discrepancy is a "**domain shift**"



An Example: From Simulations to Reality

How can we make sure neural networks are learning with the right features from simulated data?

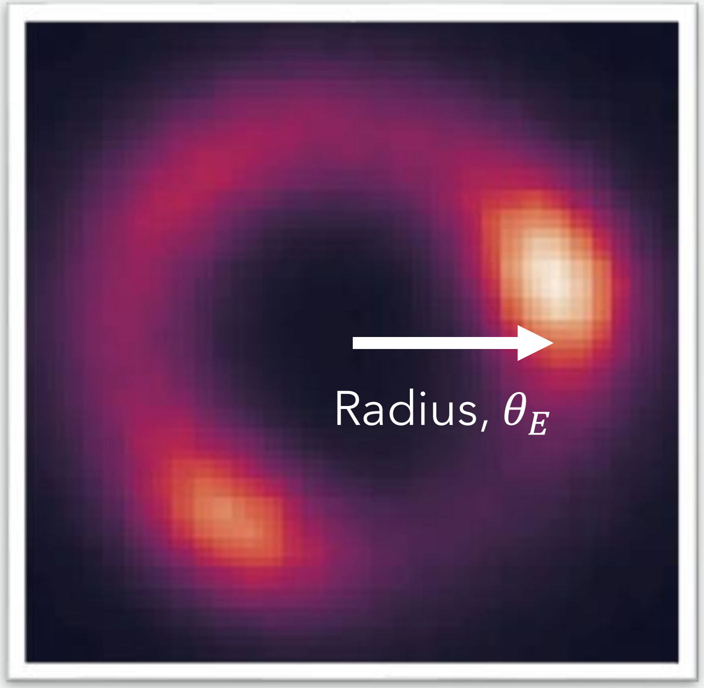
If I train on this…



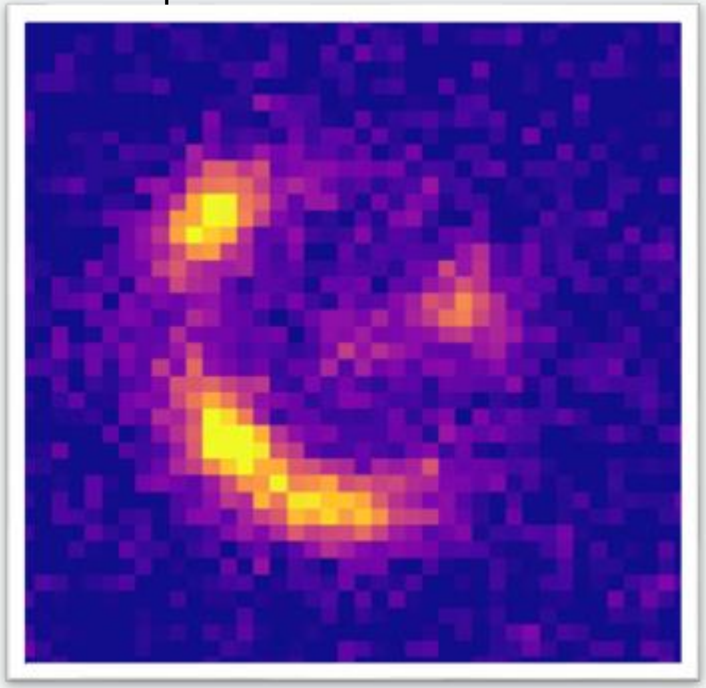
An Example: From Simulations to Reality

How can we make sure neural networks are learning with the right features from simulated data?

If I train on this…



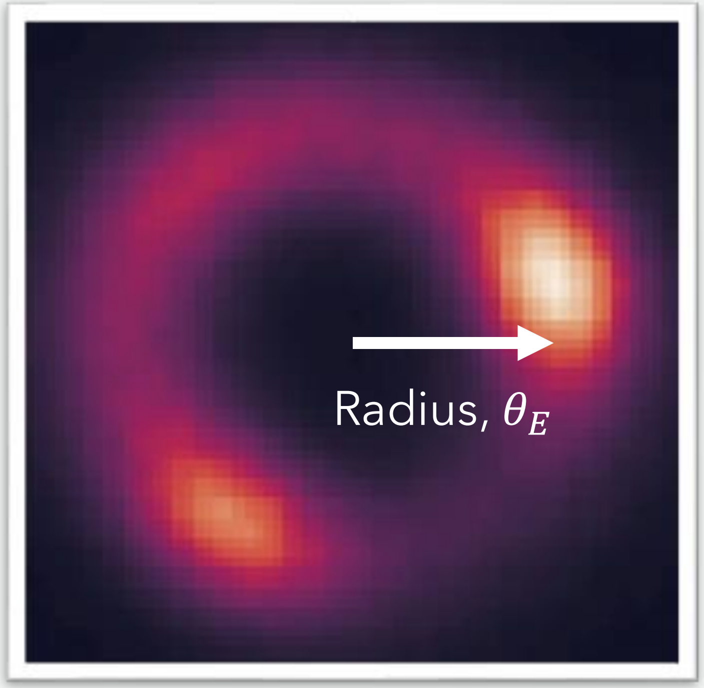
Can I predict on this?



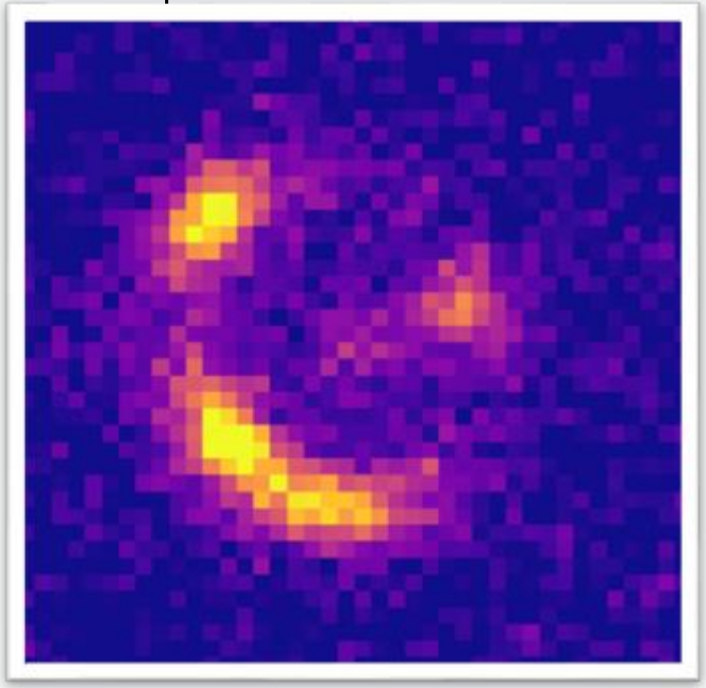
An Example: From Simulations to Reality

How can we make sure neural networks are learning with the right features from simulated data?

If I train on this…



Can I predict on this?



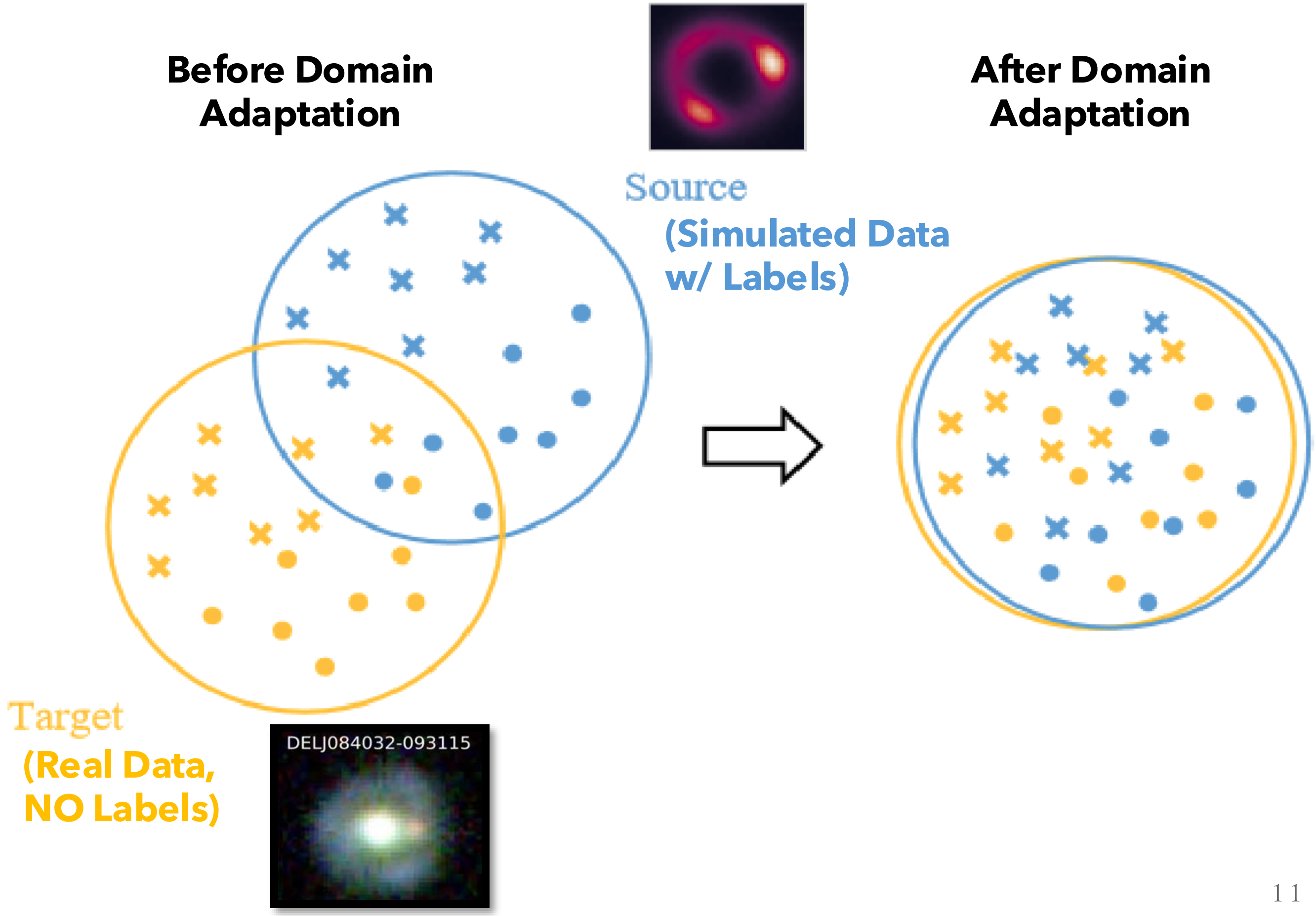
How about this?

To Get Around This, We Use Domain Adaptation

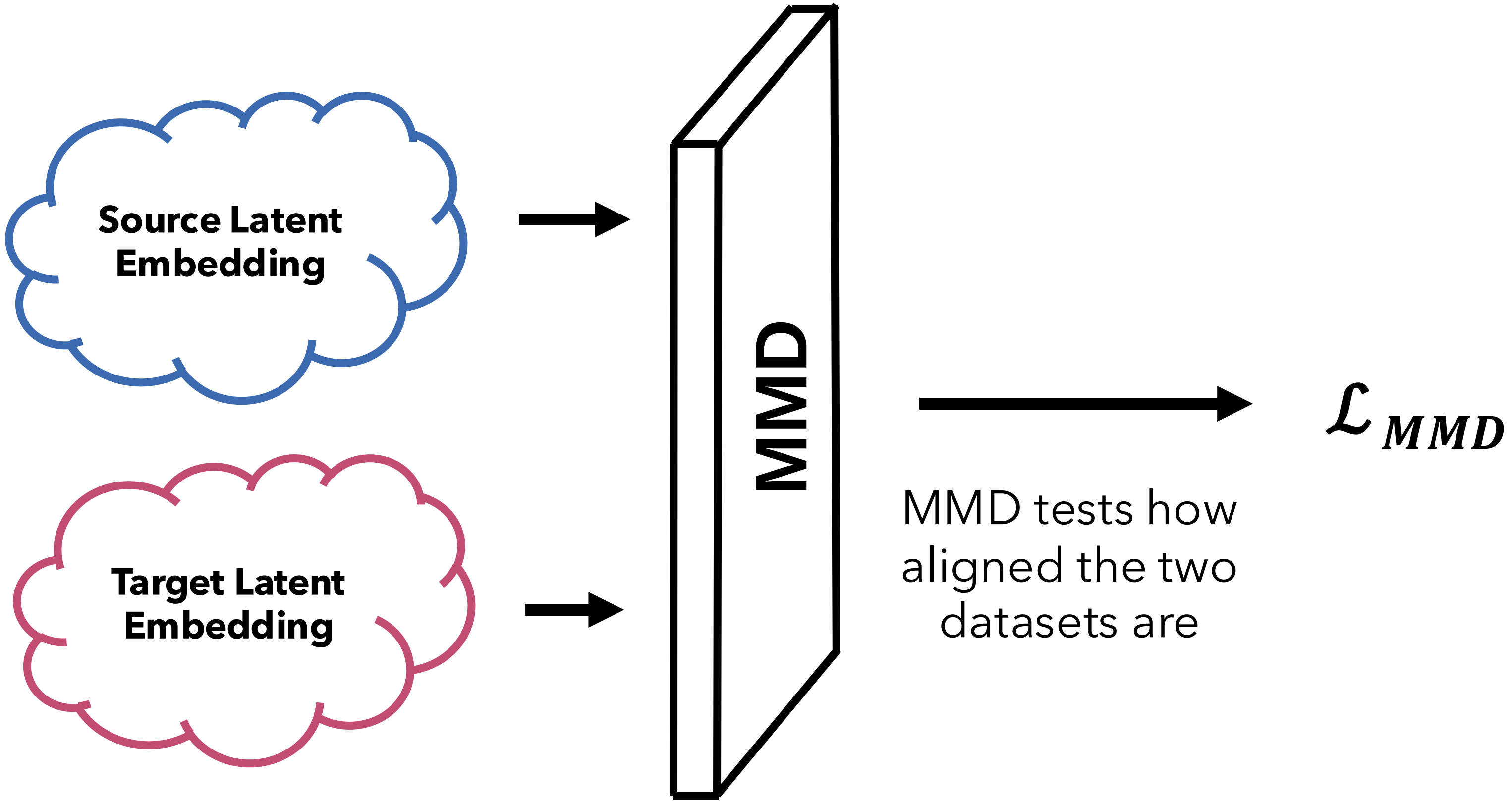
Train the network while finding invariants between the training (source) dataset and prediction (target) dataset.

These representations containing invariant and relevant features are **"latent embeddings"**.

This process is **Unsupervised!**



Maximum Mean Discrepancy (MMD)



- MMD is effectively a multi-dimensional distance aggregated between sets of vectors
- If the MMD is large, datasets are separate
- If the MMD is small, datasets are aligned
- **We optimize to reduce MMD by including it in the loss function**

Unsupervised: MMD can be calculated *without any* labels!

How About Predicting Uncertainties?

We can calculate **aleatoric** uncertainties using the beta negative log-likelihood as our loss function.

$$\mathcal{L}_{\beta-\mathrm{NLL}} := \mathop{\mathbb{E}}_{X,Y}\left[\underbrace{\lfloor\hat{\sigma}^{2\beta}(X)\rfloor}_{\text{Re-Weighting w/ }\beta}\underbrace{\left(\frac{1}{2}\log\hat{\sigma}^2(X)+\frac{(Y-\hat{\mu}(X))^2}{2\hat{\sigma}^2(X)}+\mathrm{const}\right)}_{\text{Negative Log-Likelihood}}\right]$$

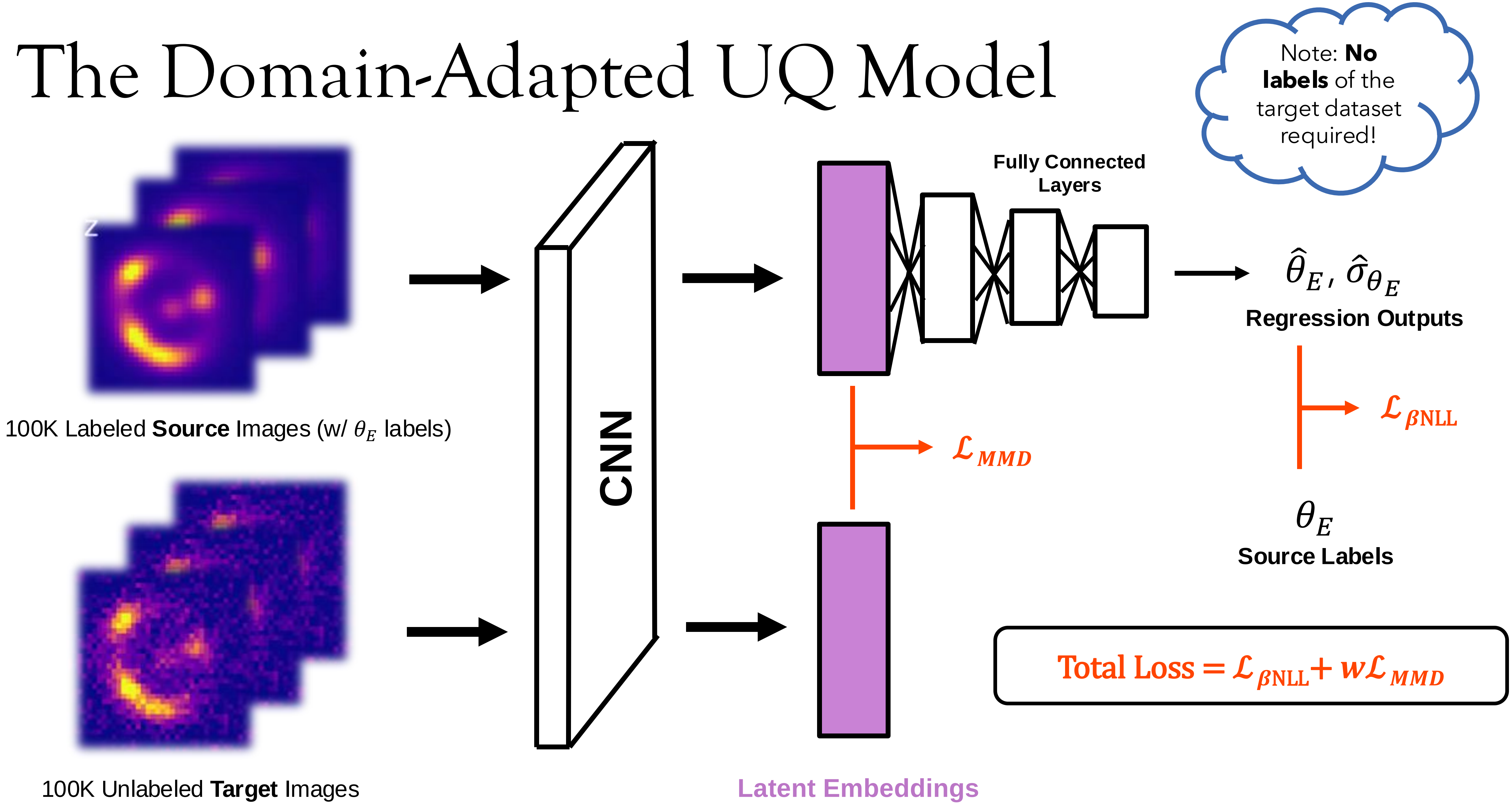
(Seitzer et. al 2022)

β varies the loss to weight accuracy vs. calibration.

$0 \longleftarrow \beta \longrightarrow 1$

Mean-squared error $\longleftarrow \mathcal{L}_{\beta\mathrm{NLL}} \longrightarrow$ **Negative Log-Likelihood**

The Domain-Adapted UQ Model



Model Results

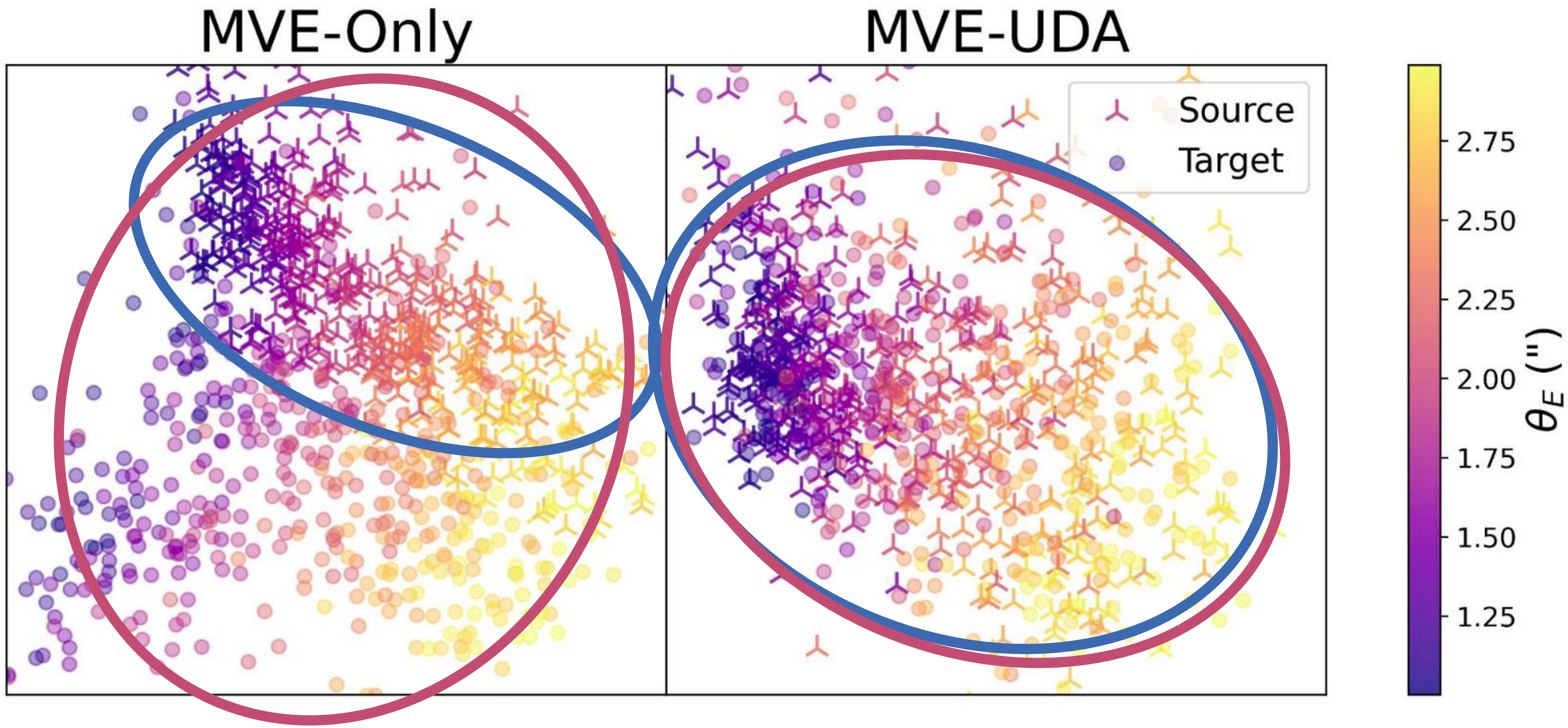
MVE-only (Not Adapted)

Vs.

MVE-UDA (Domain-Adaptive)

Alignment of Latent Embeddings w/ Domain Adaptation

Isomaps show dimensionally-reduced form of latent embeddings, while preserving distances. **Adaptation aligns embeddings in latent space.**

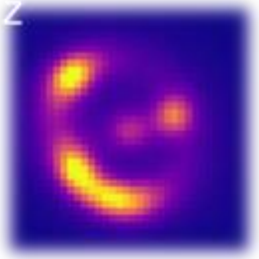


No Adaptation **w/ Adaptation**

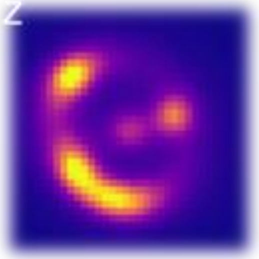
MVE-only Model Inconsistent

MVE-Only on SOURCE Dataset

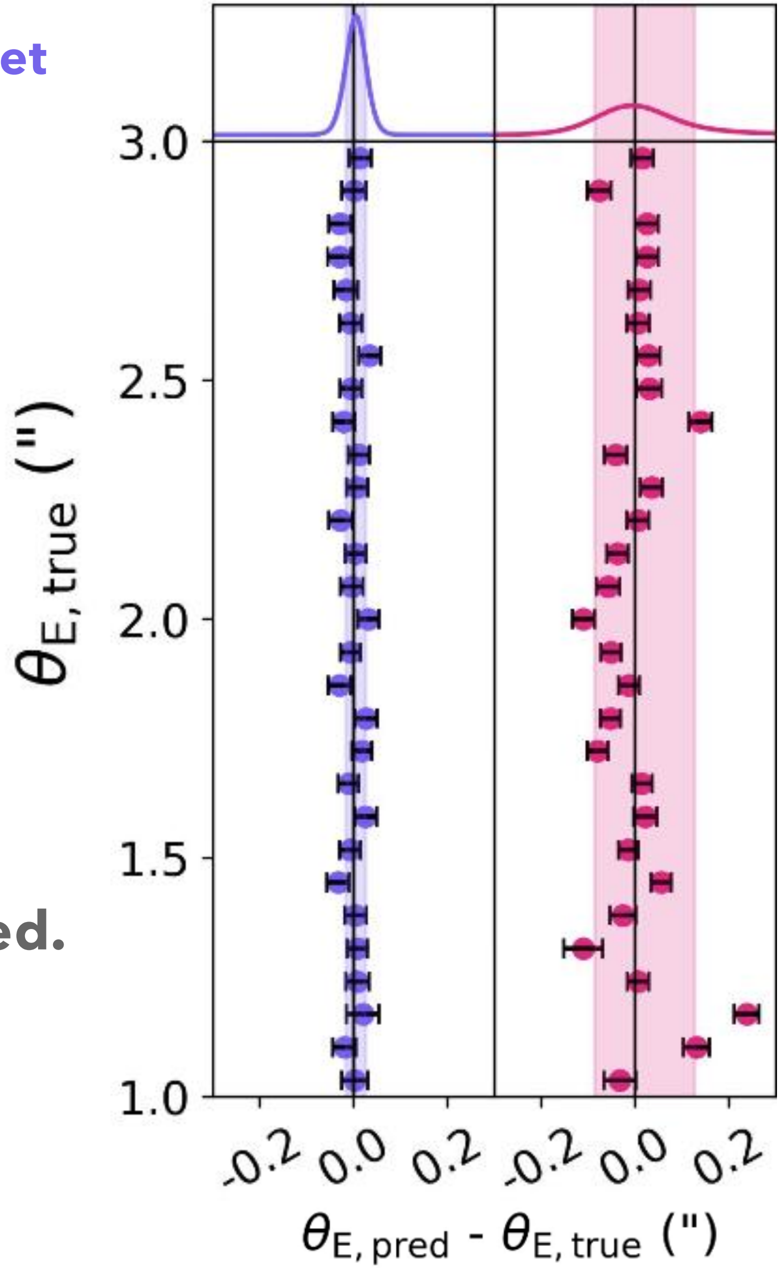
Trained on



Predicting on

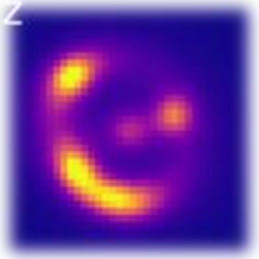


Performs well, as expected.

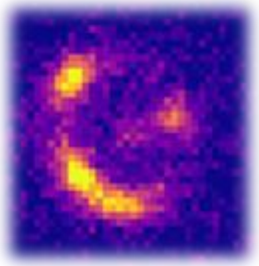


MVE-Only on TARGET Dataset

Trained on



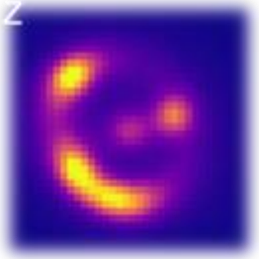
Predicting on

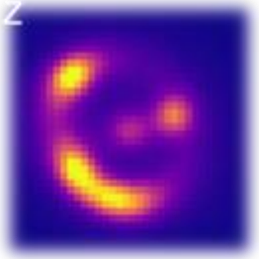


Poorly calibrated & inaccurate.

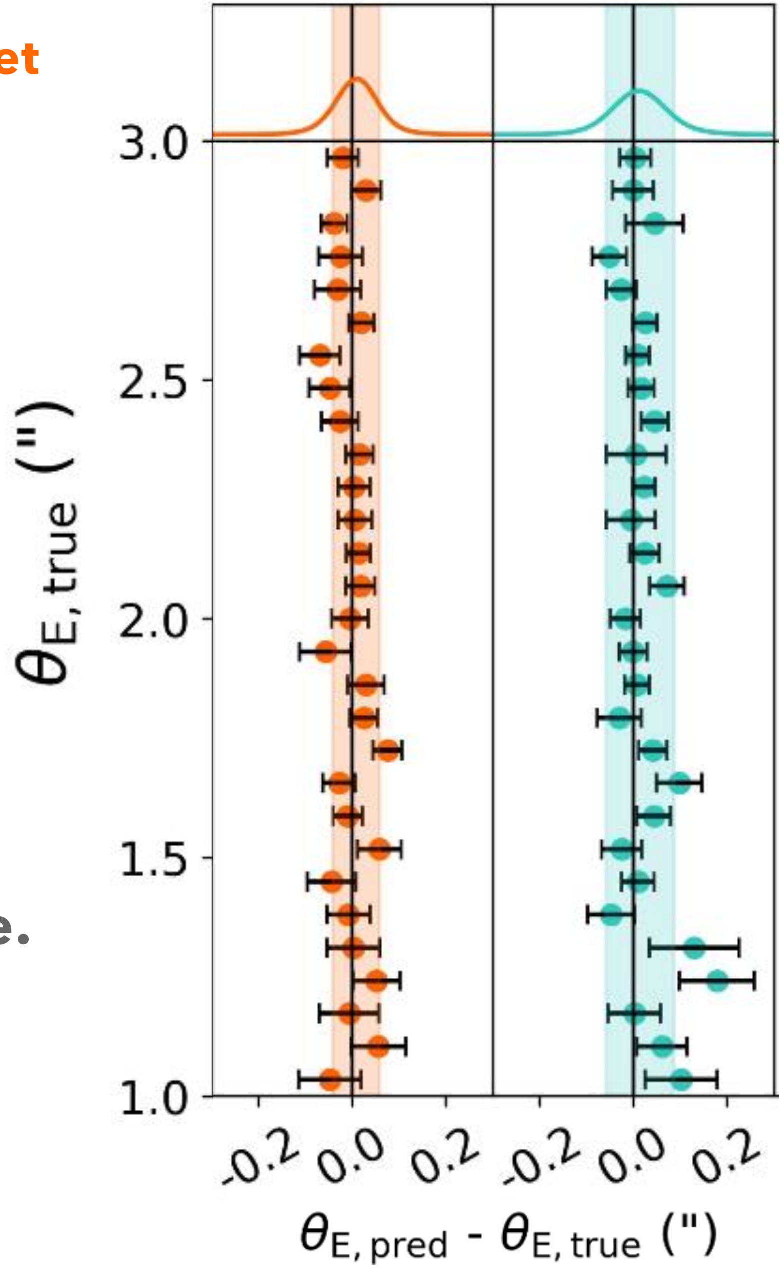
MVE-UDA Model Consistent

MVE-UDA on SOURCE Dataset

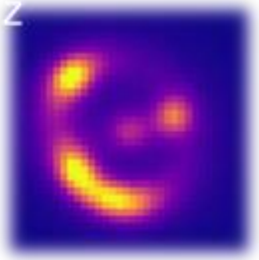
Trained on 

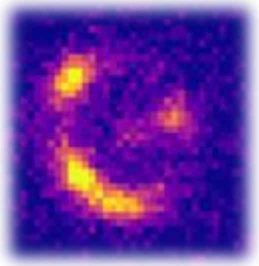
Predicting on 

Well calibrated, accurate.



MVE-UDA on TARGET Dataset

Trained on 

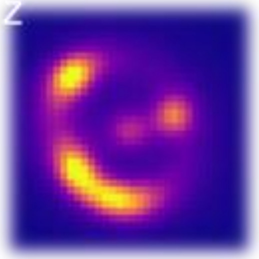
Predicting on 

Maintains accuracy, calibration.

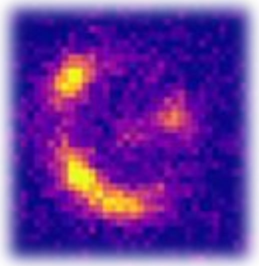
Comparing Performance on Target Datasets

MVE-Only on TARGET Dataset

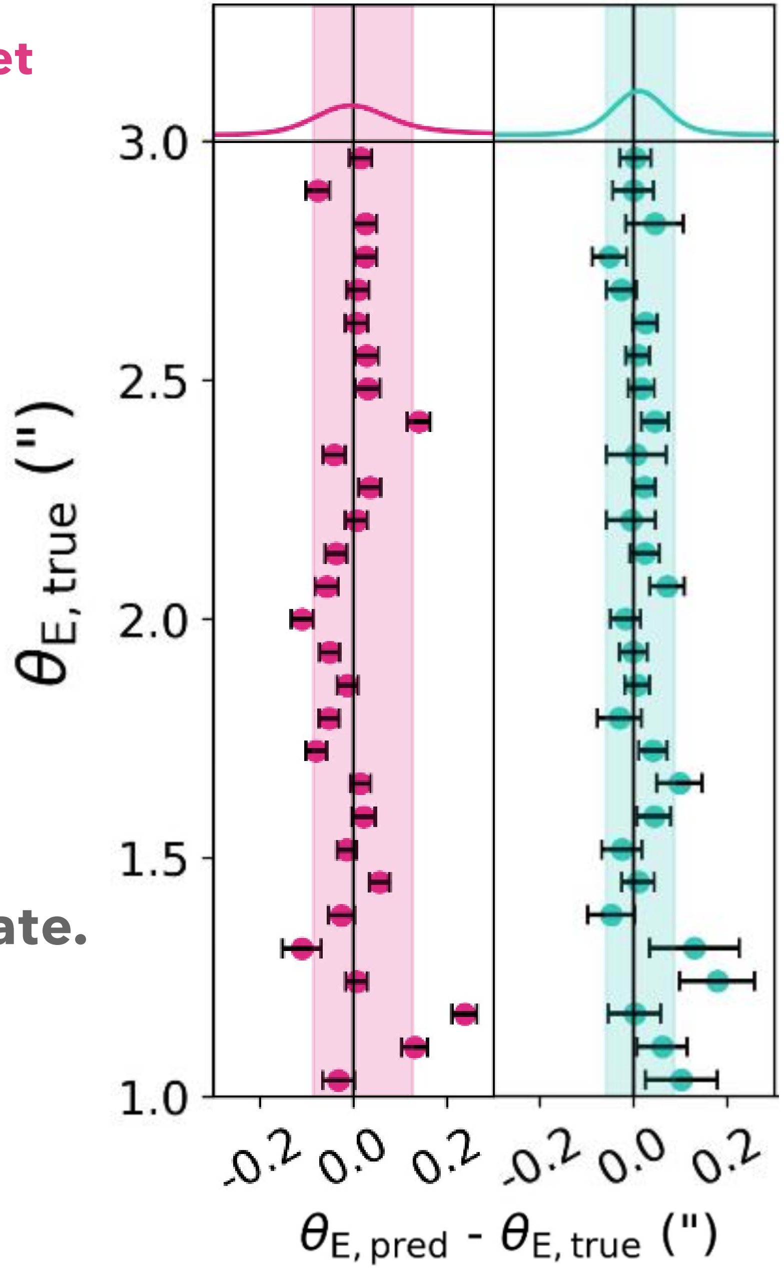
Trained on



Predicting on

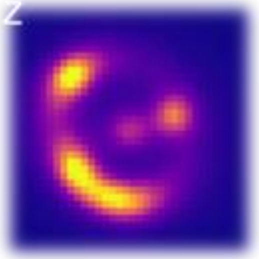


Badly calibrated & inaccurate.

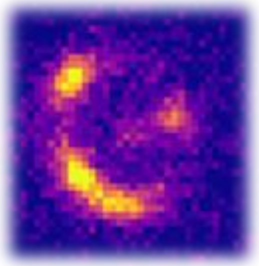


MVE-UDA on TARGET Dataset

Trained on



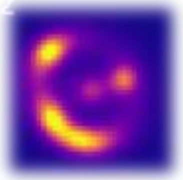
Predicting on



Accurate, well calibrated.

All Mean and Variance Predictions

All models trained on the source dataset

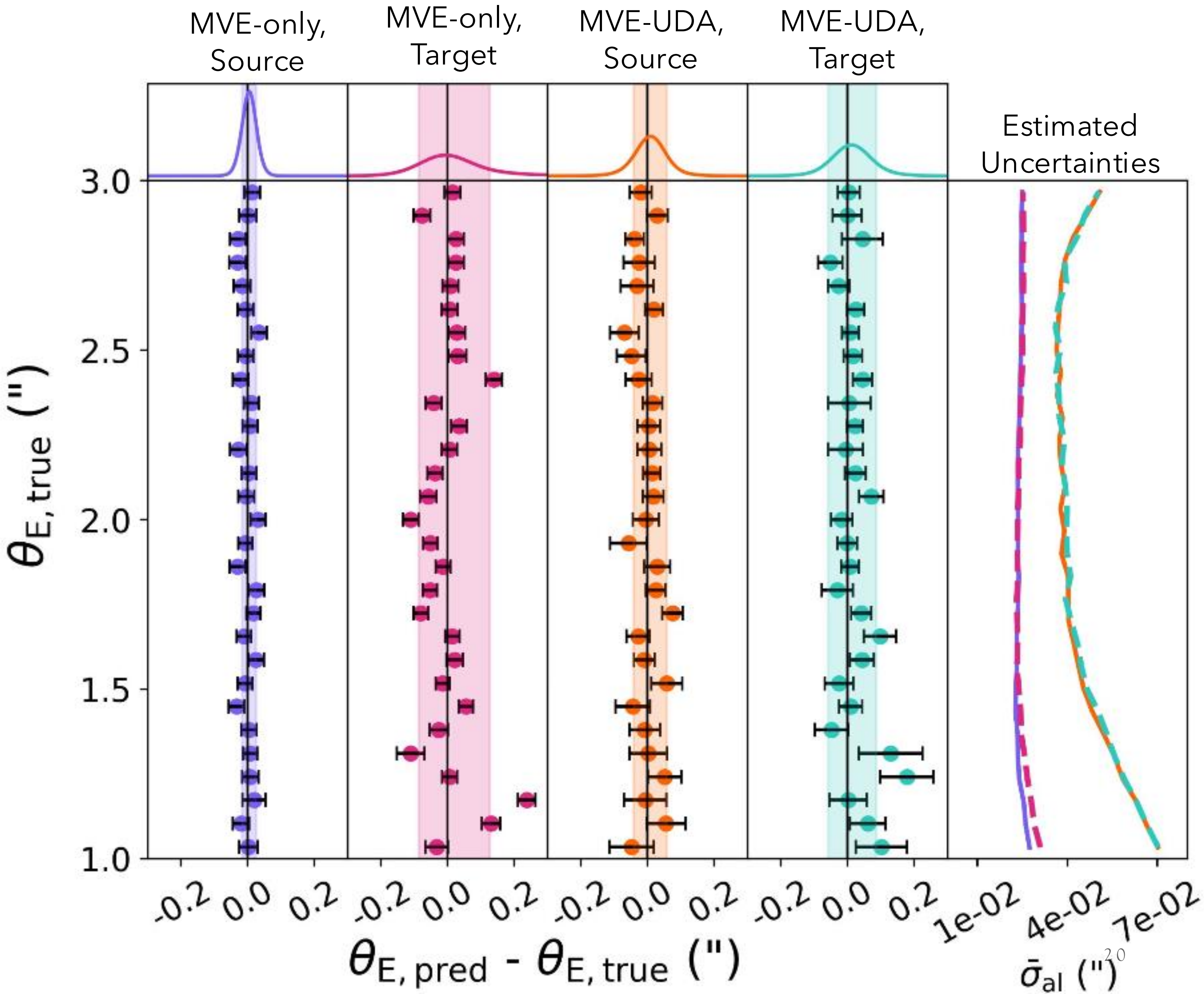


MVE-only model inconsistent on source vs. target

MVE-UDA model consistent across source and target

MVE-UDA is more accurate and calibrated than the target

Model variances are higher at edges of distribution



MVE-UDA allows well-calibrated uncertainty on domain-shifted data.

The MVE-only model is significantly **overconfident** on the target dataset

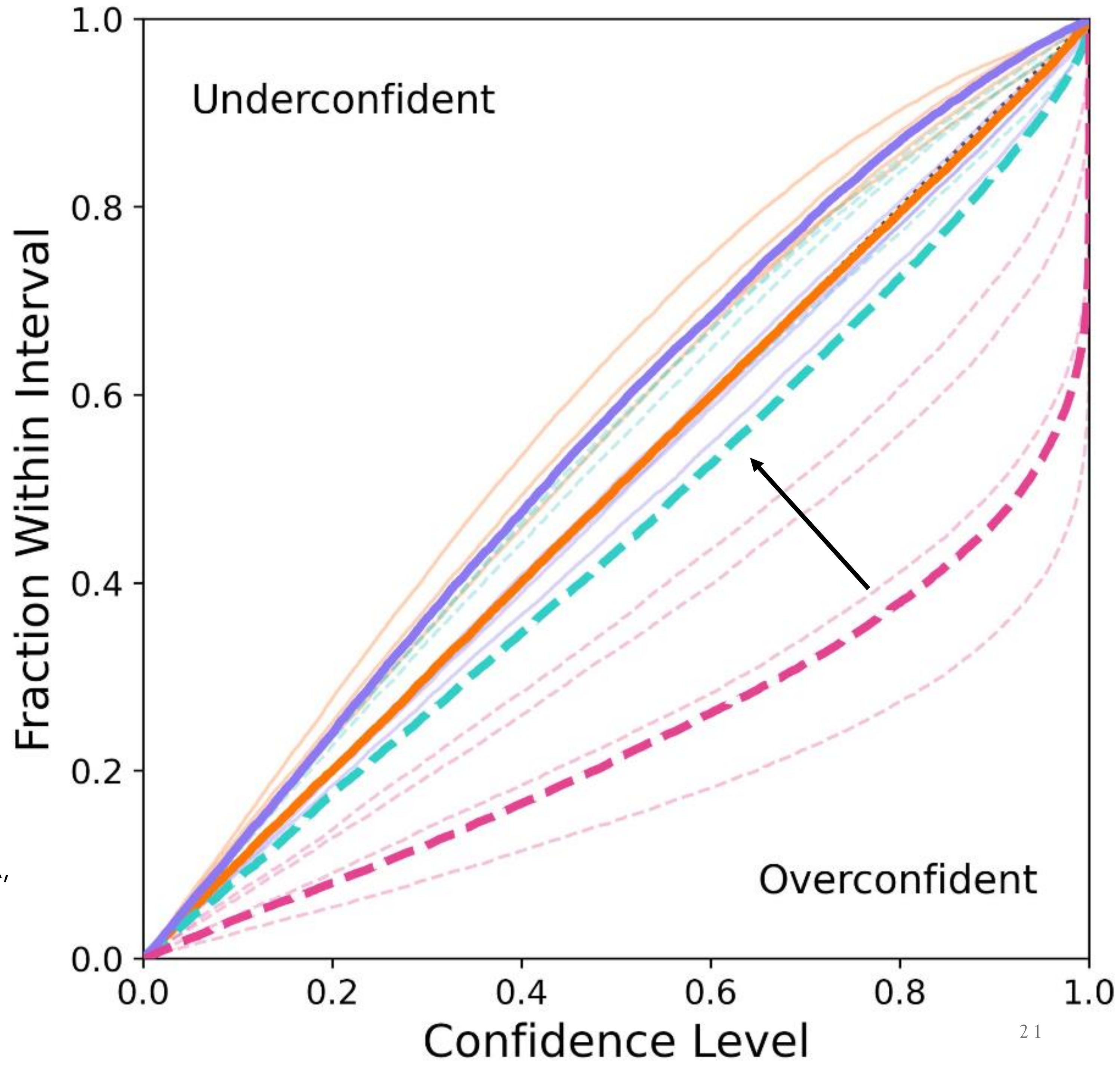
These **results hold** across varying seed initializations (faded lines)

MVE-only, Source

MVE-only, Target

MVE-UDA, Source

MVE-UDA, Target



Conclusions

- On target (noisy) data, we found that the DA model

  *Was **more accurate** than the MVE-only model*

  *Was significantly **better calibrated** than the MVE-only model*

  *Was **more consistent** across varying weight initializations, MVE-only was unpredictable*

- DA performs slightly worse on training (source) data, to trade-off for better performance on target data

- This scheme of DA + UQ is a general concept that can be applied beyond strong lensing.

EXTRA SLIDES

Table 1: Prior distributions of the simulation parameters for training and test sets.

Parameter		Prior
Lens light profile		
Einstein radius	θ_{E} ($''$)	$\mathcal{U}(1.0, 3.0)$
Sérsic index	n	$\mathcal{U}(2.0, 5.0)$
Scale radius	R ($''$)	$\mathcal{U}(1.0, 2.5)$
Eccentricity	$\{e_{\mathrm{l},1}, e_{\mathrm{l},2}\}$	$\mathcal{U}(-0.2, 0.2)$
External shear	$\{\gamma_1, \gamma_2\}$	$\mathcal{U}(-0.05, 0.5)$
Source light profile		
Sérsic index	n	$\mathcal{U}(2.0, 4.0)$
Scale radius	R ($''$)	$\mathcal{U}(0.5, 1.0)$
Eccentricity	$\{e_{\mathrm{s},1}, e_{\mathrm{s},2}\}$	$\mathcal{U}(-0.2, 0.2)$
Relative angular positions	$\{x, y\}$ ($''$)	$\mathcal{U}(-0.5, 0.5)$

Table 3: The architecture of the MVE network. The first column lists the layer type, the second lists the dimensionality of the output from that layer, and the third column lists the parameters of that layer; k is the kernel size, and s is the stride. The final layer outputs the mean and variance.

Layer	Output shape	Parameters
Conv2d	[-1, 8, 40, 40]	$k = 3, s = 1$
BatchNorm2d	[-1, 8, 40, 40]	$k = 3, s = 1$
MaxPool2d	[-1, 8, 20, 20]	$k = 2, s = 2$
Conv2d	[-1, 16, 20, 20]	$k = 3, s = 1$
BatchNorm2d	[-1, 16, 20, 20]	$k = 3, s = 1$
MaxPool2d	[-1, 16, 20, 20]	$k = 2, s = 2$
Conv2d	[-1, 32, 10, 10]	$k = 3, s = 1$
BatchNorm2d	[-1, 32, 10, 10]	$k = 3, s = 1$
MaxPool2d	[-1, 32, 5, 5]	$k = 2, s = 2$
Linear	[-1, 128]	-
Linear	[-1, 32]	-
Linear	[-1, 2]	-

Table 4: Mean residual $\langle \delta\theta_{\mathrm{E}} \rangle$, mean aleatoric uncertainty $\langle \sigma_{\mathrm{al}} \rangle$, mean correlation coefficient $\langle R^2 \rangle$, and mean NLL loss $\langle \mathcal{L}_{\beta-\mathrm{NLL}} \rangle$ across each data set for each model, MVE-only, MVE-UDA.

		MVE-only		MVE-UDA	
Metric	Seed	Source	Target	Source	Target
Residual: $\langle \delta\theta_{\mathrm{E}} \rangle$	56	0.0164	0.0693	0.0358	0.0436
	11	0.0149	0.0287	0.0389	0.0425
	31	0.0201	0.0585	0.0386	0.0461
	6	0.0150	0.0818	0.0484	0.0510
	63	0.0174	0.0240	0.0452	0.0551
Uncertainty: $\langle \sigma_{\mathrm{al}} \rangle$	56	0.0243	0.0253	0.0489	0.0503
	11	0.0180	0.0179	0.0602	0.0599
	31	0.0269	0.0239	0.0634	0.0634
	6	0.0192	0.0199	0.0678	0.0678
	63	0.0203	0.0205	0.0628	0.0628
Correlation: $\langle R^2 \rangle$	56	0.9986	0.9642	0.9924	0.9835
	11	0.9988	0.9939	0.9917	0.9897
	31	0.9979	0.9727	0.9922	0.9886
	6	0.9988	0.9418	0.9880	0.9861
	63	0.9984	0.9968	0.9889	0.9832
NLL Loss: $\langle \mathcal{L}_{\beta-\mathrm{NLL}} \rangle$	56	-3.3603	4.5586	-2.6600	-2.4204
	11	-3.4737	-1.0705	-2.5098	-2.4385
	31	-3.1443	15503.4180	-2.4316	-2.2854
	6	-3.4925	25.4278	-2.2687	-2.2070
	63	-3.2745	-2.6643	-2.2982	-2.0623