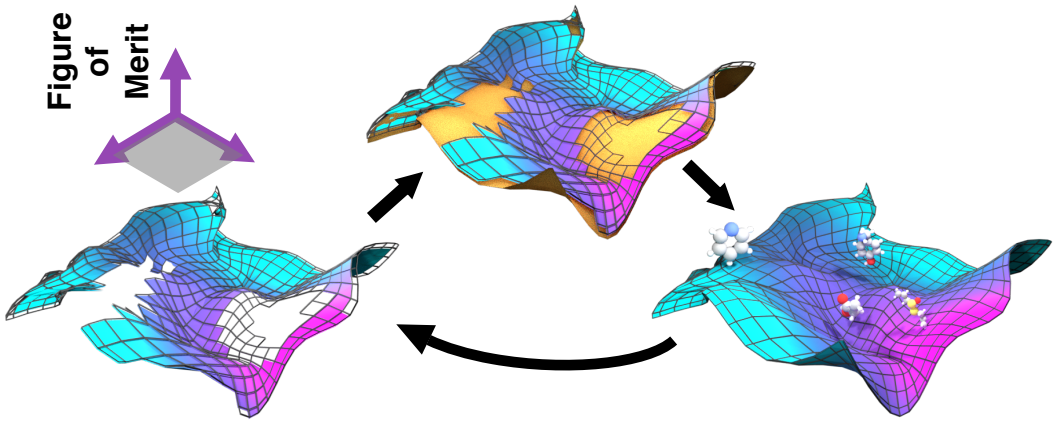
Goals for Today:

Paradigms with examples:

Contemporary Topics:

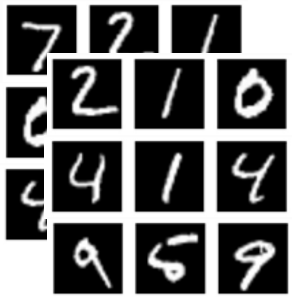


Tutorial Example:

COC(=O)CC(=C) ⟶ T_g, (E_{coh}, …)

Generative Models Are Neat

Autoencoders are "self-supervised" machine-learning models for generating low-dimensional data encodings



MNIST handwritten digit dataset (70k images)

convolutional models or gated recurrent units

convolutional models or gated recurrent units

1) Convert to tensor

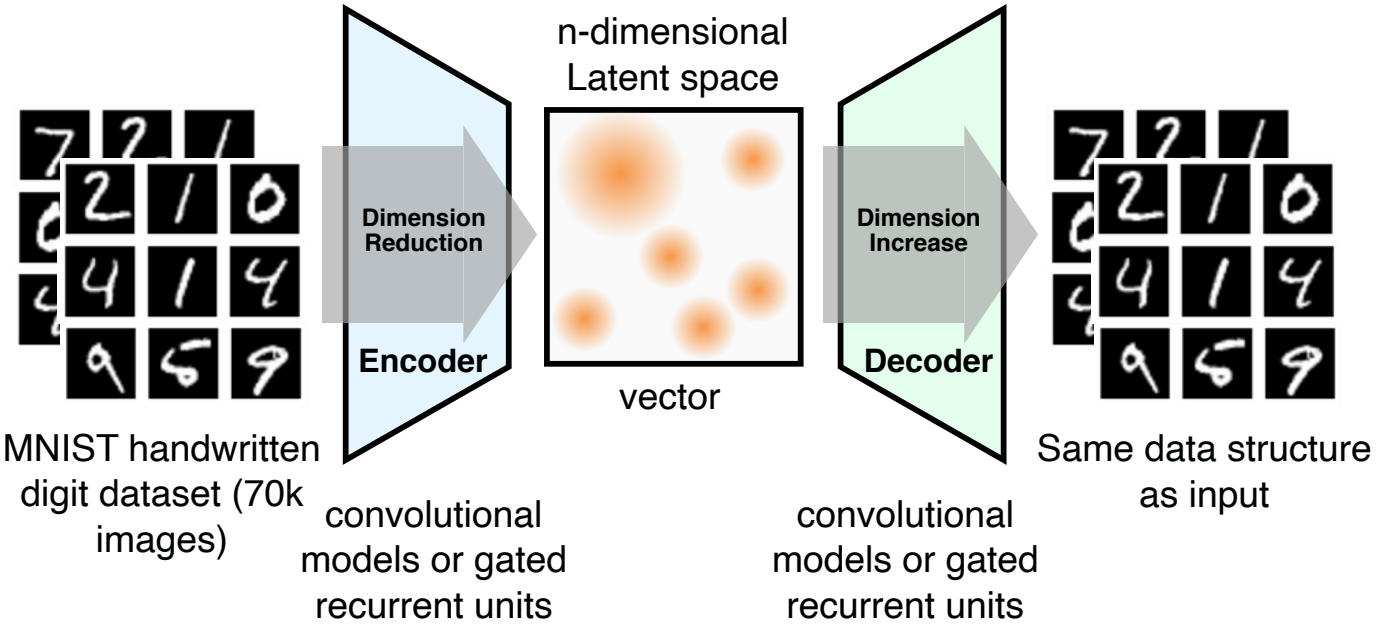
2) Encode: $f(\phi_{in}) = \mathbf{r}$

3) Decode: $g(\mathbf{r}) = \phi_{out}$

4) Train on reconstruction loss

Generative Models Are Neat

Autoencoders are "self-supervised" machine-learning models for generating low-dimensional data encodings



1) Convert to tensor

variational

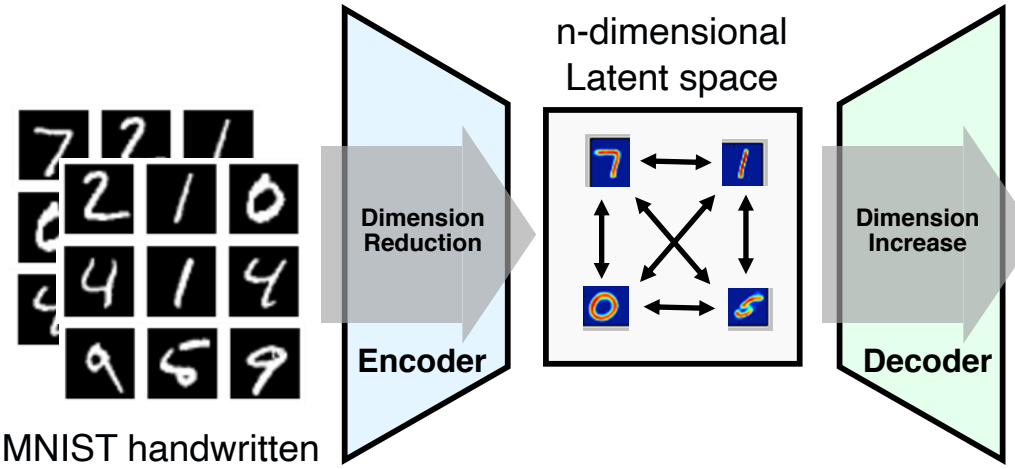
2) Encode: $f(\phi_{in}) = \mathbf{r}$ $\quad f(\phi_{in}) = (\boldsymbol{\mu}, \boldsymbol{\sigma})$

3) Decode: $g(\mathbf{r}) = \phi_{out}$ $\quad g(\boldsymbol{\mu}, \boldsymbol{\sigma}) = \phi_{out}$

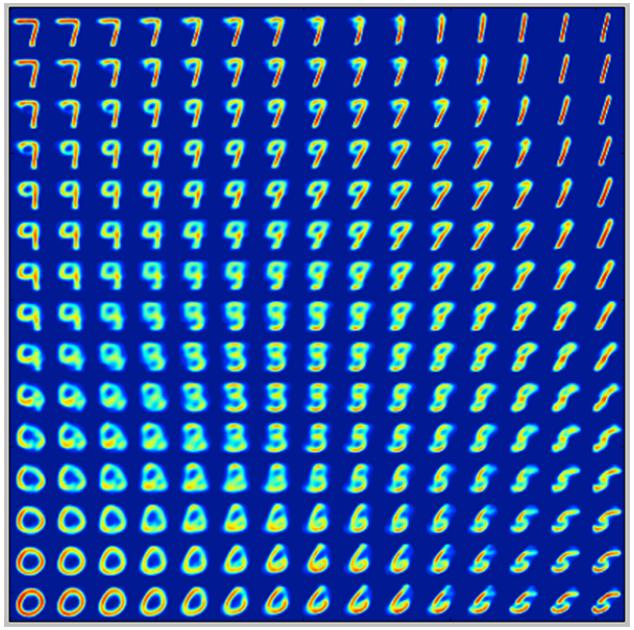
4) Train on reconstruction loss

Generative Models Are Neat

Autoencoders are "self-supervised" machine-learning models for generating low-dimensional data encodings



MNIST handwritten digit dataset (70k images)



Corners are the only "real" images

After training, the latent space can be explored and decoded to "generate" new data [i.e., gives us access to **p(X|Y)**]

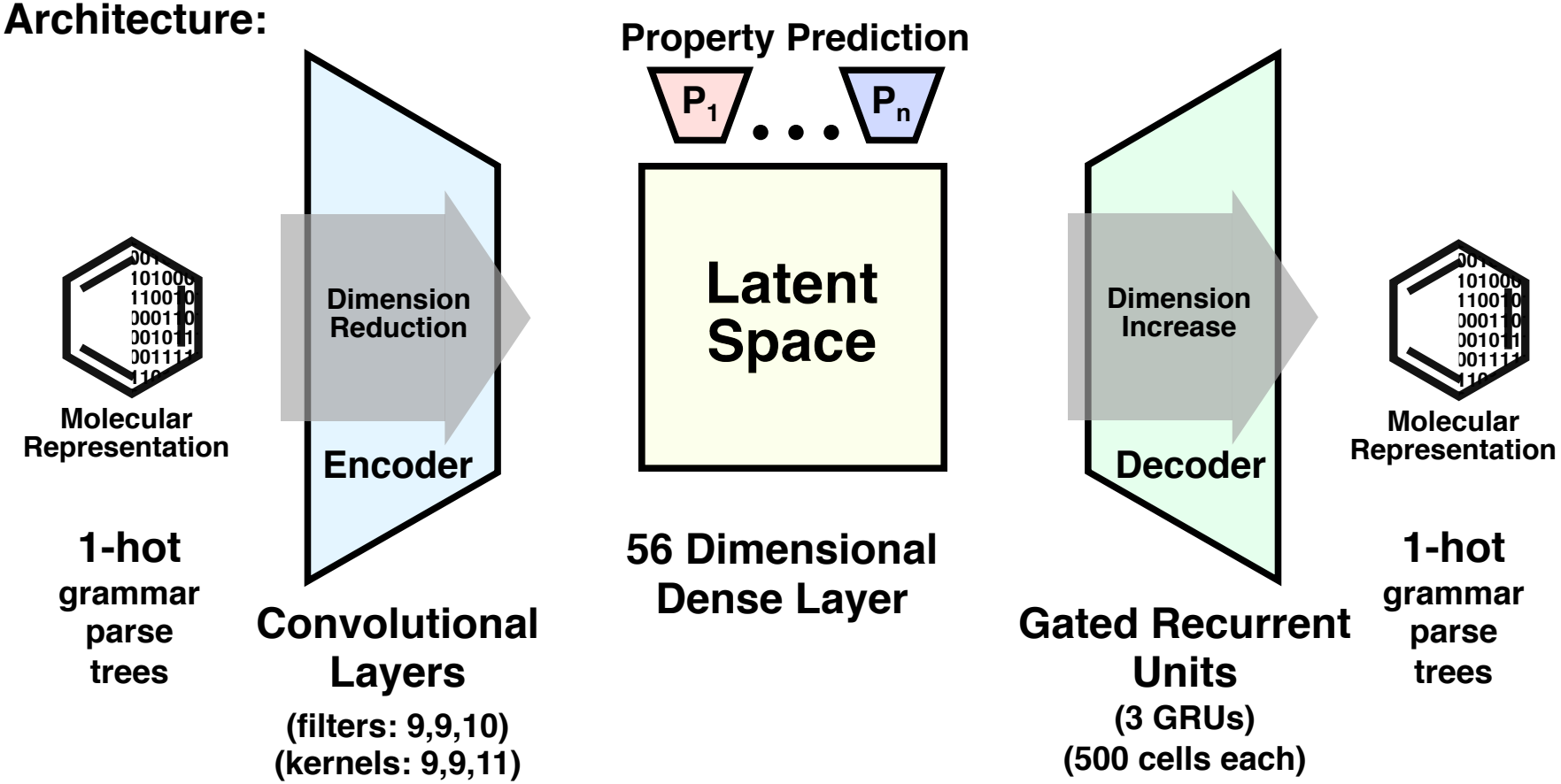
Transfer Learning for Generative Chemical Models

Data: 134k QM9 B3LYP/6-31G* calculations (80:10:10 split), Extended with xTB E_g

Ramakrishnan, R.; Dral, P. O.; Rupp, M.; von Lilienfeld, O. A. *Scientific Data* **2014**, *1* (1).

Bannwarth, C.; Ehlert, S.; Grimme, S. *J. Chem. Theory Comput.* **2019**, *15* (3), 1652–1671.

Architecture:



Iovanac, N. C.; Savoie, B. M. "Simpler is Better: How Linear Prediction Tasks Improve Transfer Learning in Chemical Autoencoders." J. Phys. Chem. A. **2020**

Transfer Learning for Generative Chemical Models

Data: 134k QM9 B3LYP/6-31G* calculations (80:10:10 split), Extended with xTB E_g

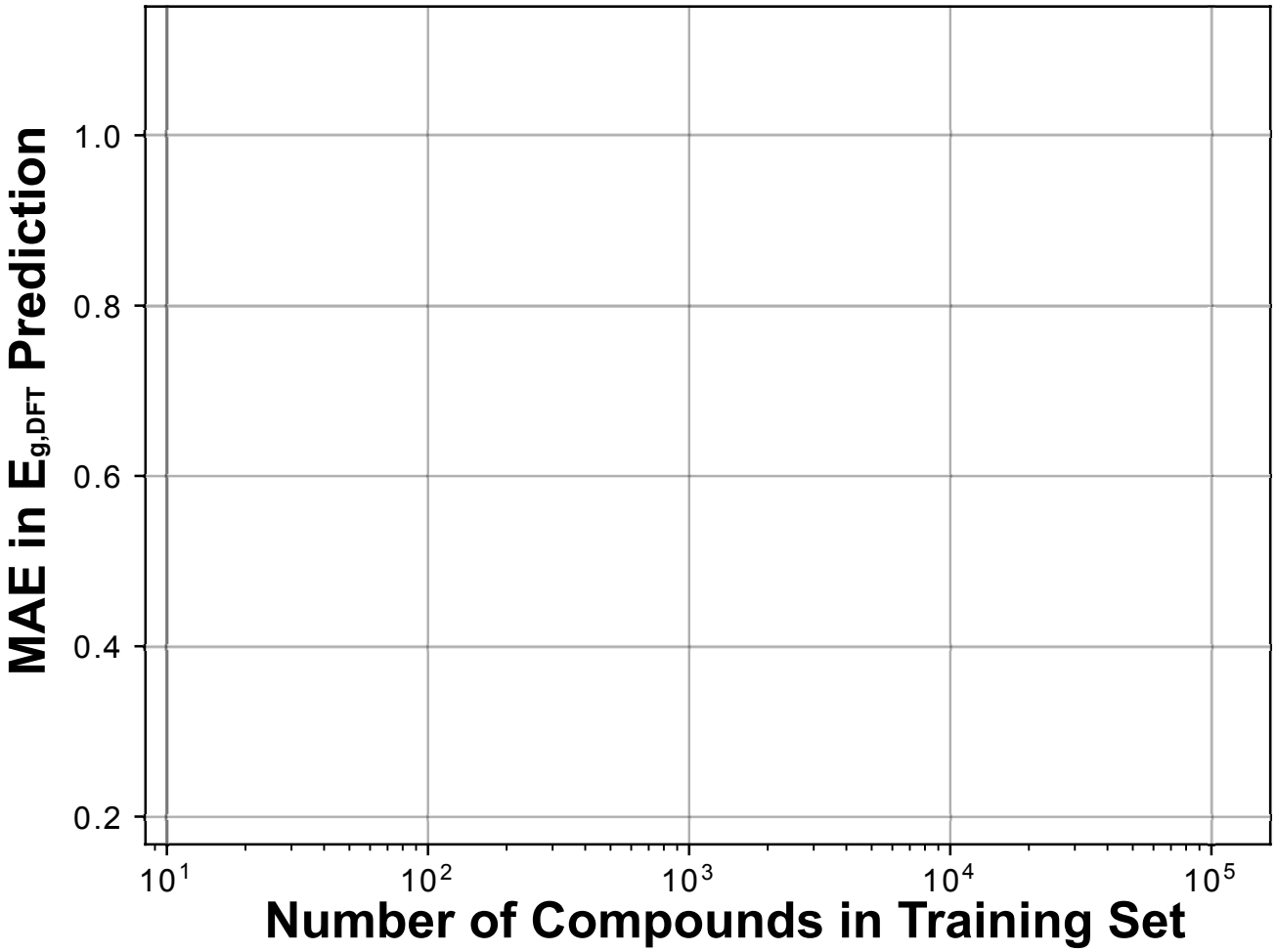
This is effectively a multitask transfer learning activity, **but for a model that is capable of generating compounds.**

First we want to show that transfer learning works for **prediction** from auto-encoder based models.

Second, we want to see if TL positively affects the **generative** properties of the model.

Iovanac, N. C.; Savoie, B. M. "Simpler is Better: How Linear Prediction Tasks Improve Transfer Learning in Chemical Autoencoders." J. Phys. Chem. A. **2020**

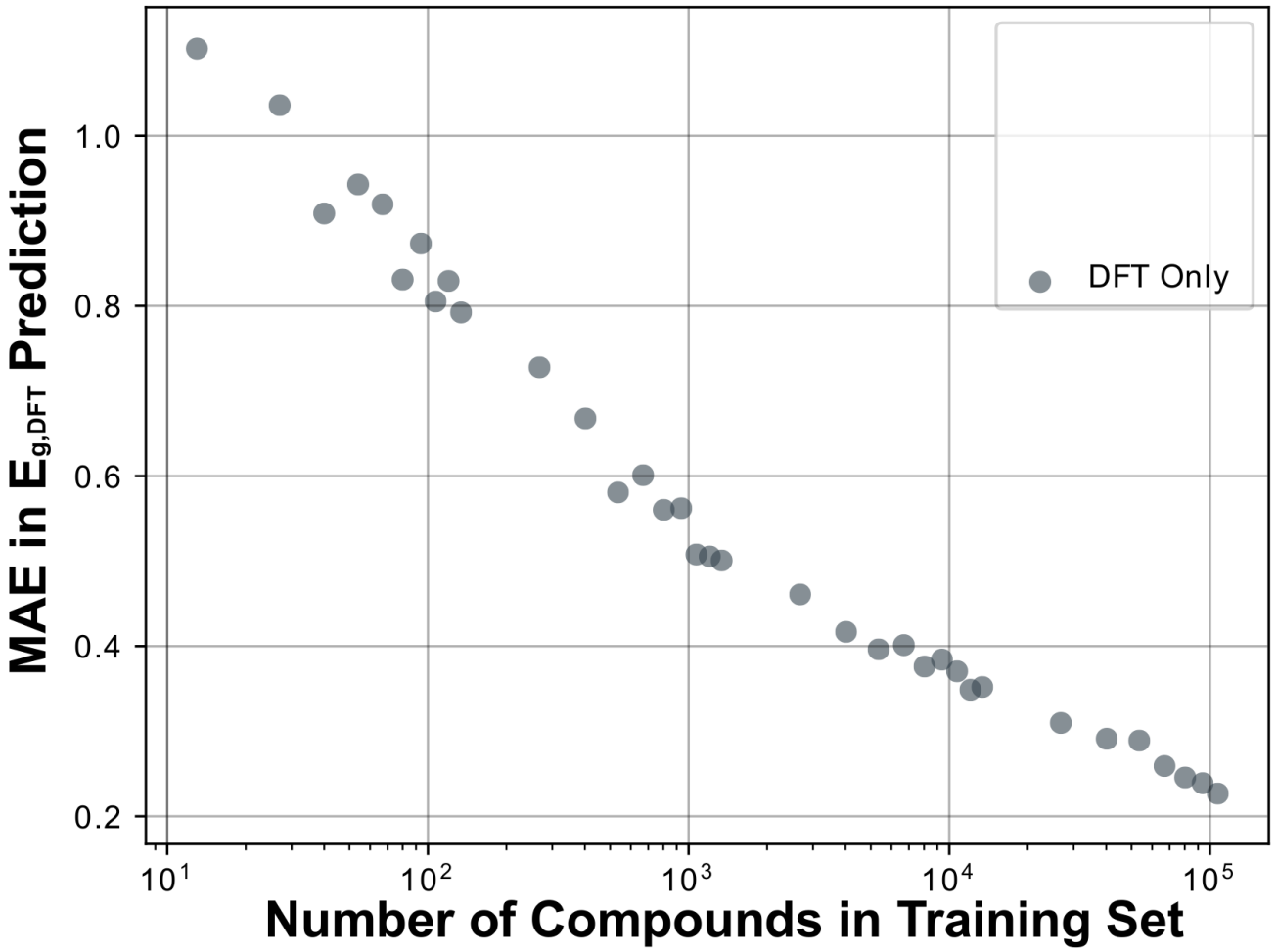
Transfer Learning on the Molecular Bandgap



Data Rich: XTB bandgap prediction (<1s per compound)

Data Scarce: DFT bandgap prediction (hours per compound)

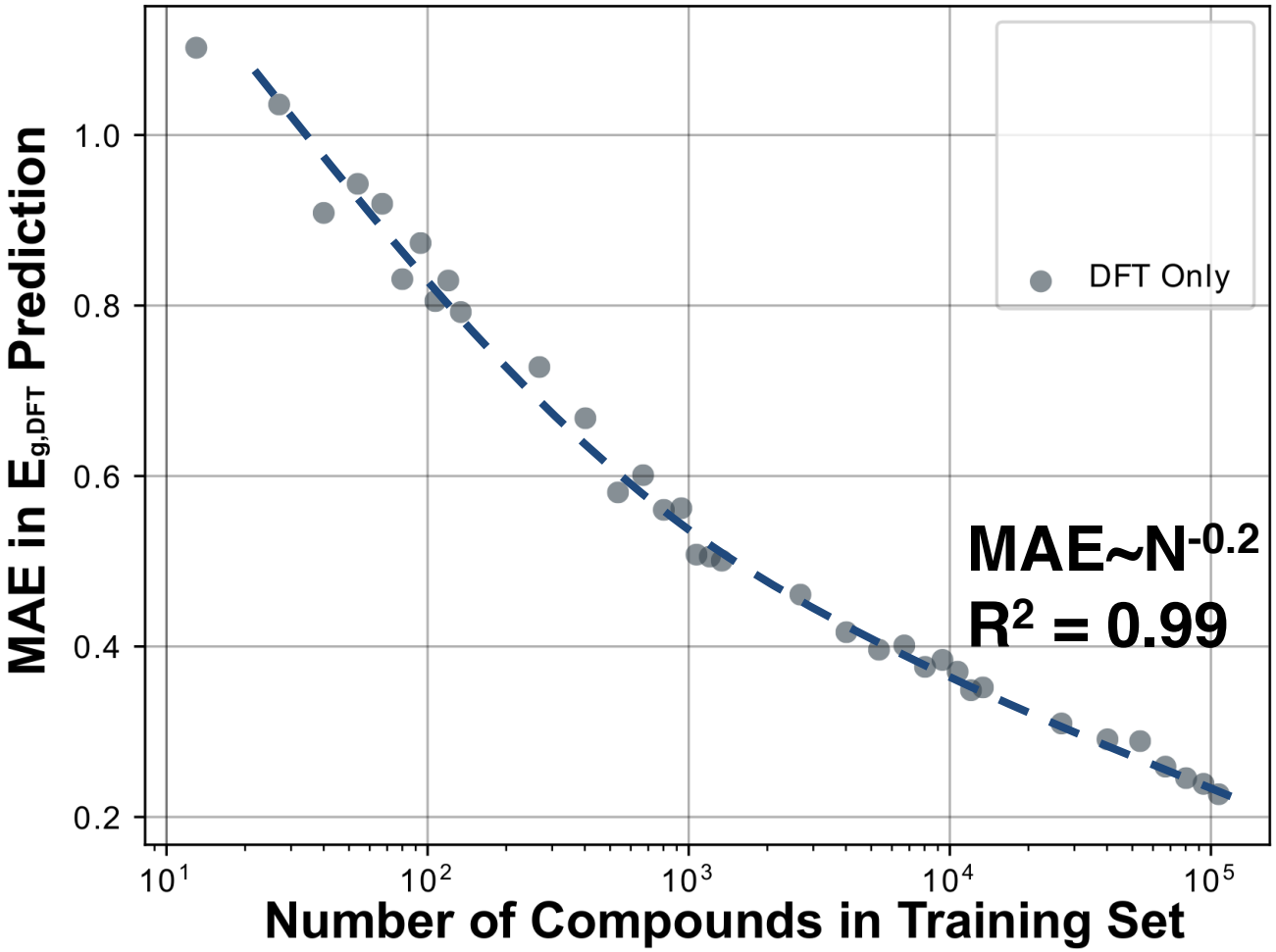
Transfer Learning on the Molecular Bandgap



Data Rich: XTB bandgap prediction (<1s per compound)

Data Scarce: DFT bandgap prediction (hours per compound)

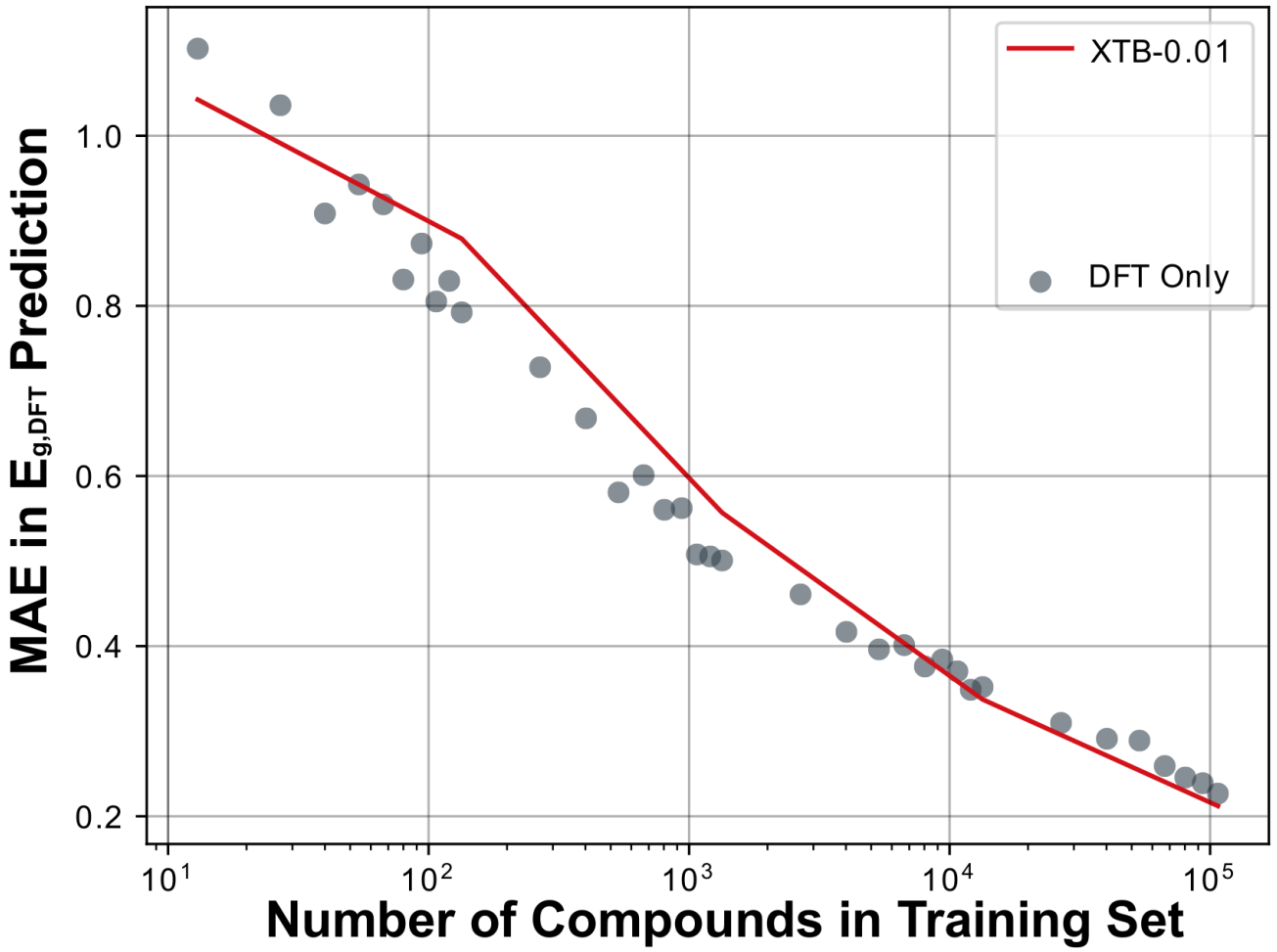
Transfer Learning on the Molecular Bandgap



Data Rich: XTB bandgap prediction (<1s per compound)

Data Scarce: DFT bandgap prediction (hours per compound)

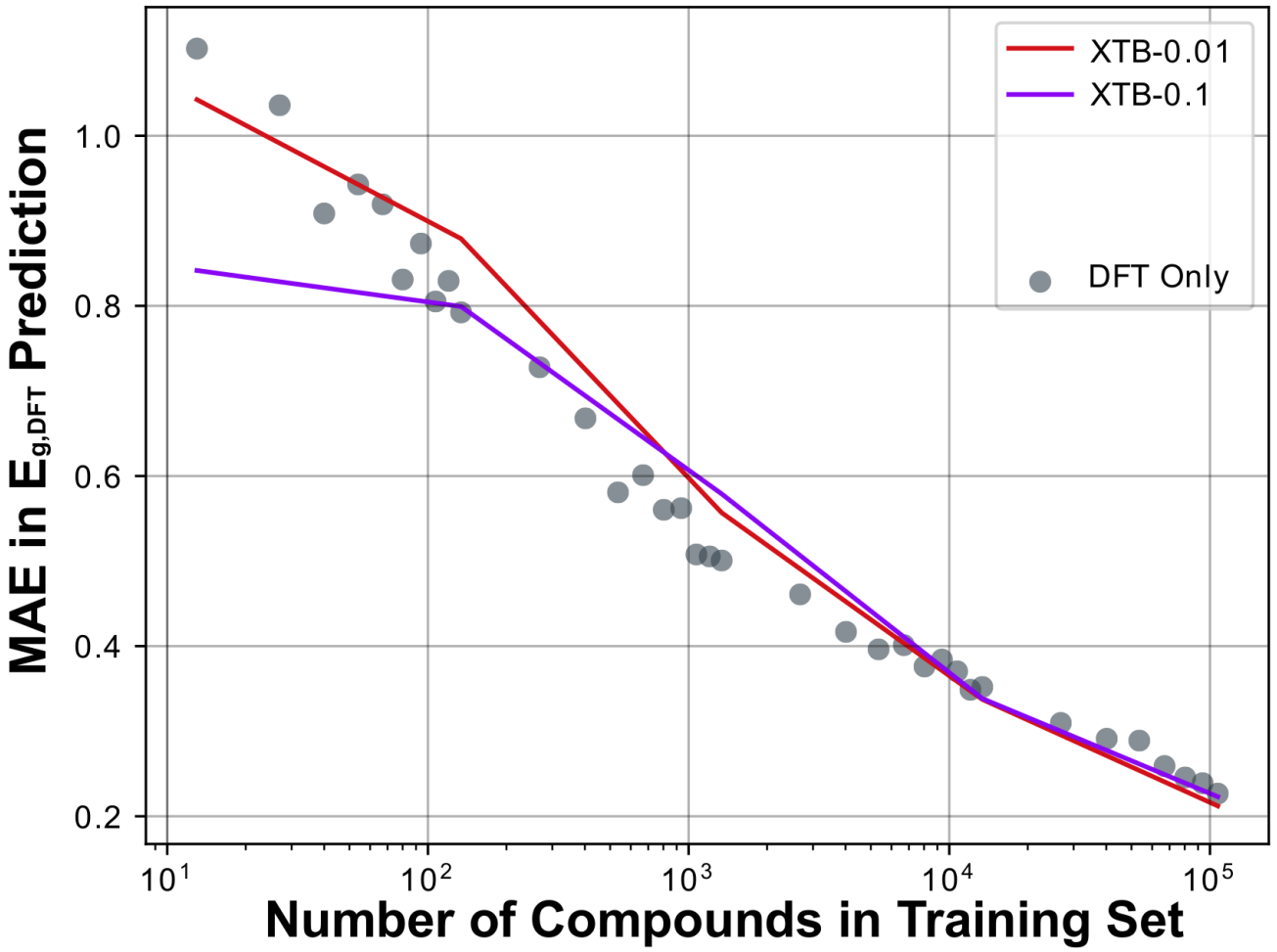
Transfer Learning on the Molecular Bandgap



Data Rich: XTB bandgap prediction (<1s per compound)

Data Scarce: DFT bandgap prediction (hours per compound)

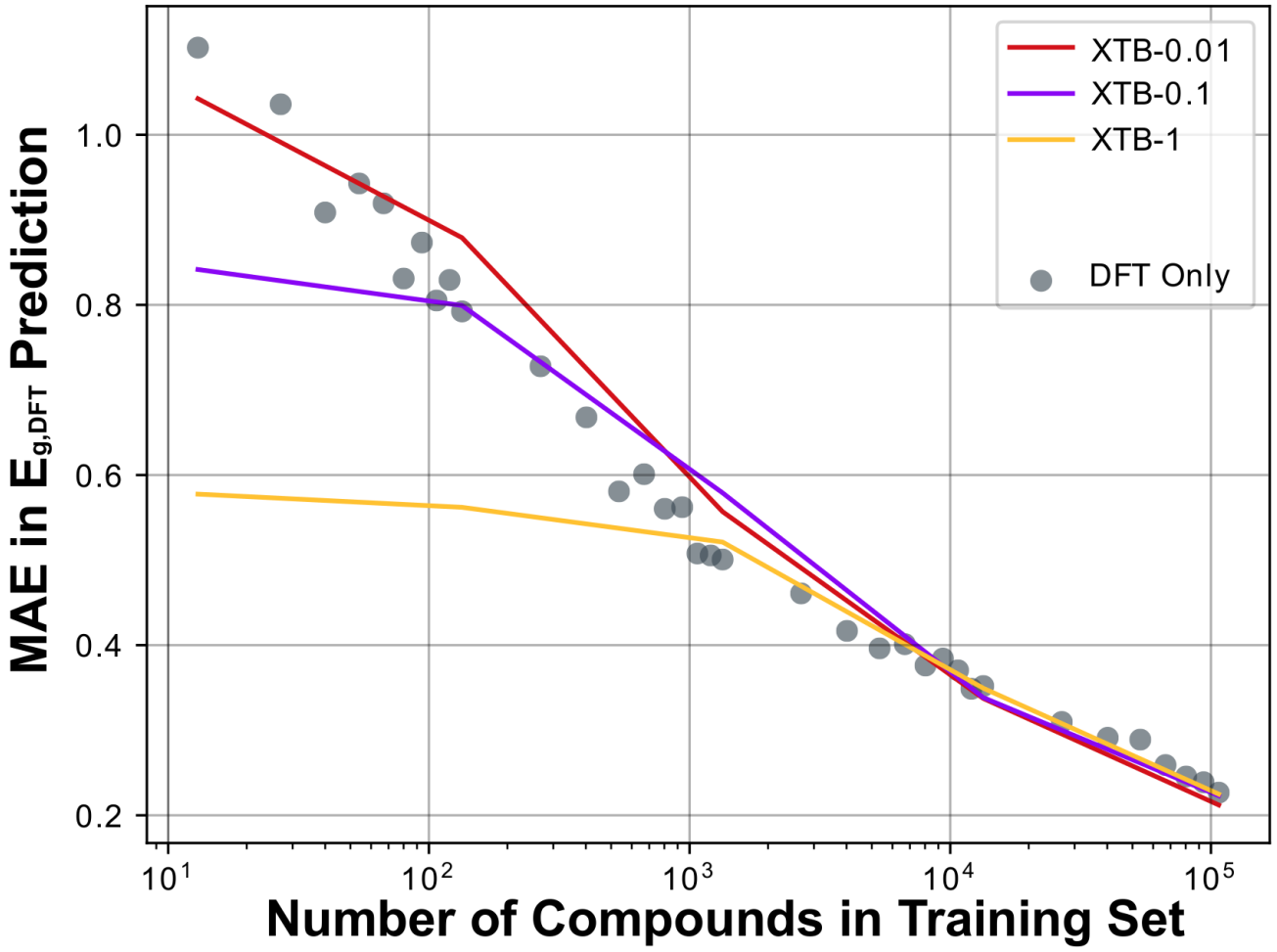
Transfer Learning on the Molecular Bandgap



Data Rich: XTB bandgap prediction (<1s per compound)

Data Scarce: DFT bandgap prediction (hours per compound)

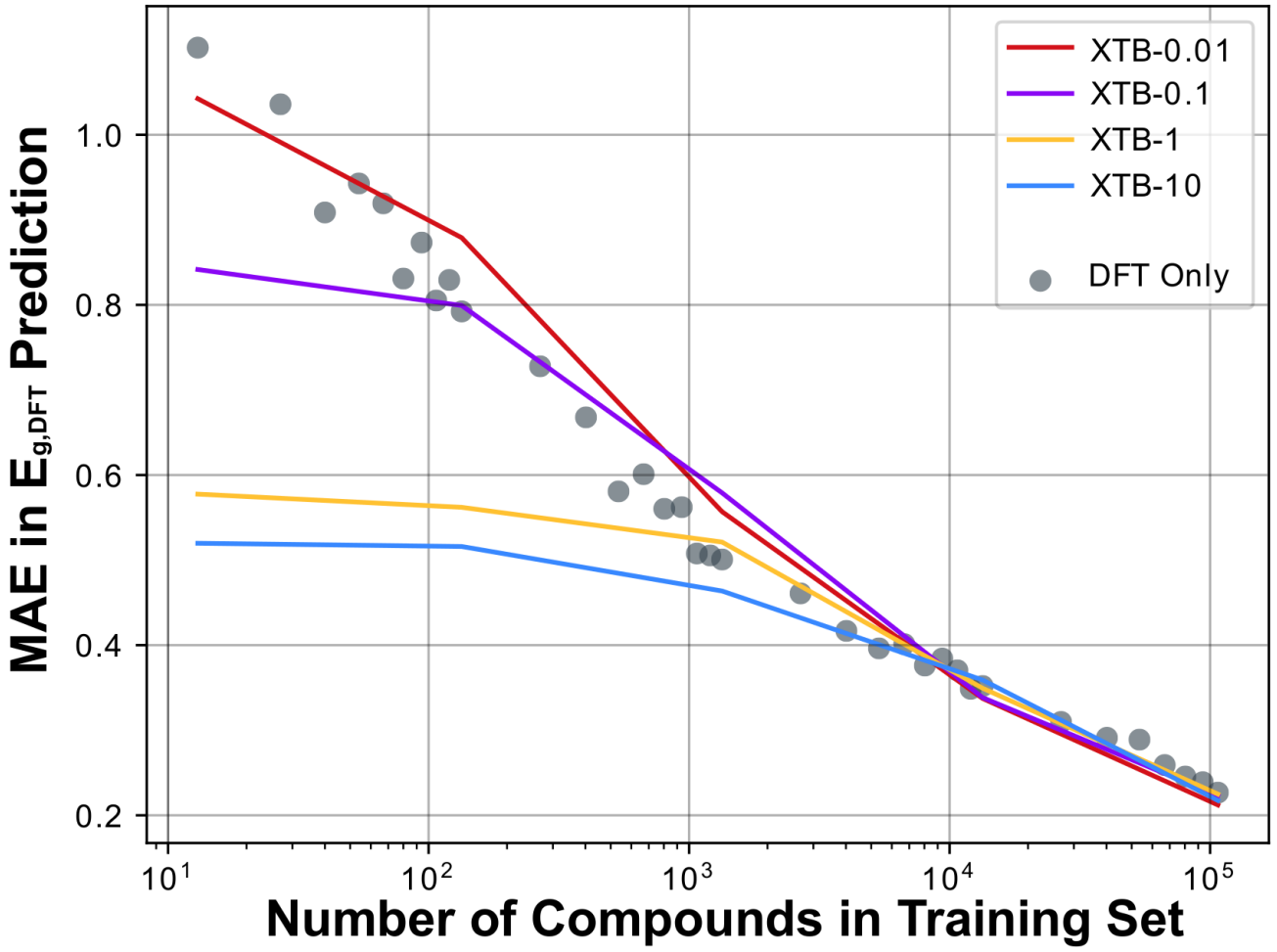
Transfer Learning on the Molecular Bandgap



Data Rich: XTB bandgap prediction (<1s per compound)

Data Scarce: DFT bandgap prediction (hours per compound)

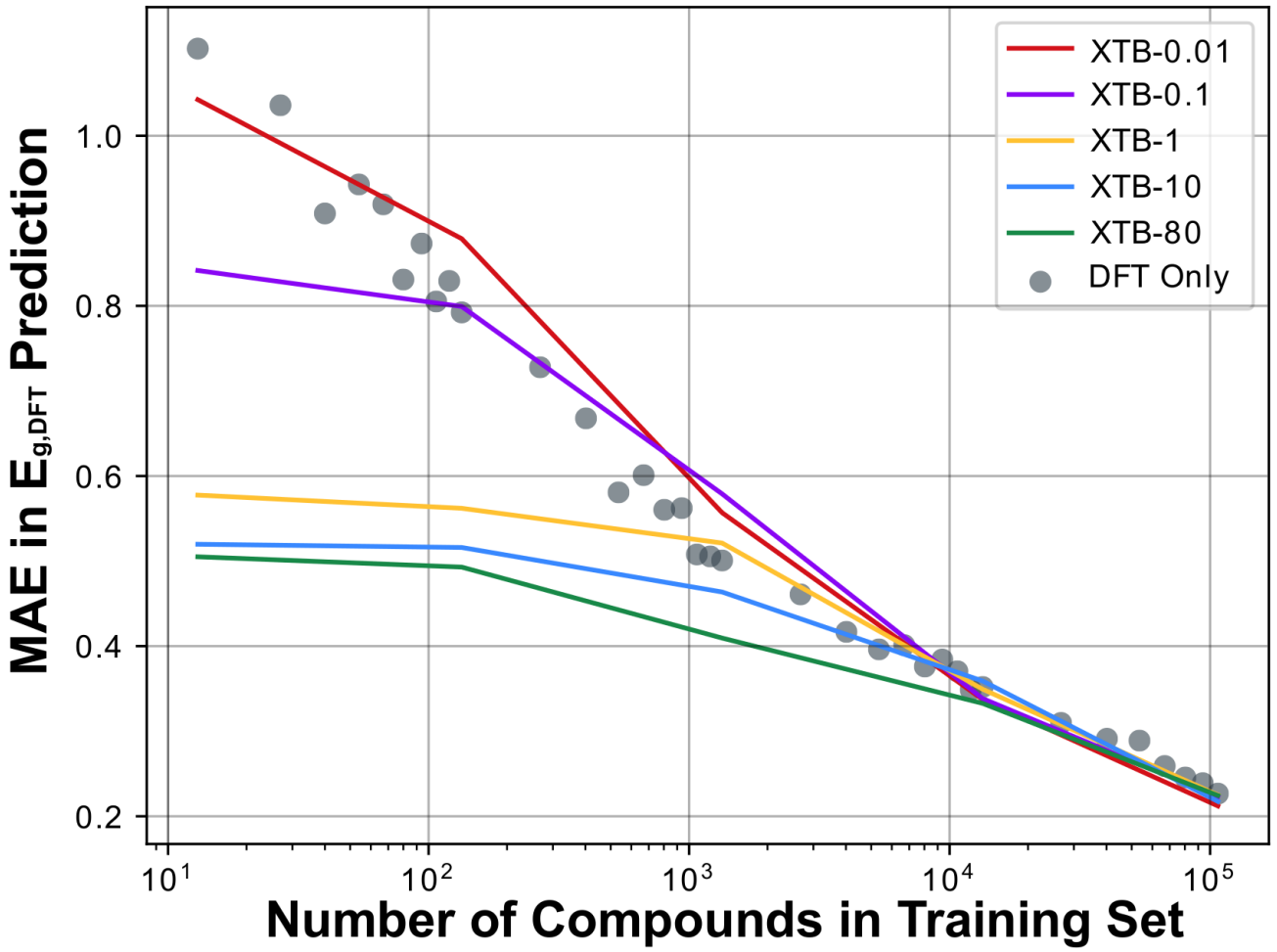
Transfer Learning on the Molecular Bandgap



Data Rich: XTB bandgap prediction (<1s per compound)

Data Scarce: DFT bandgap prediction (hours per compound)

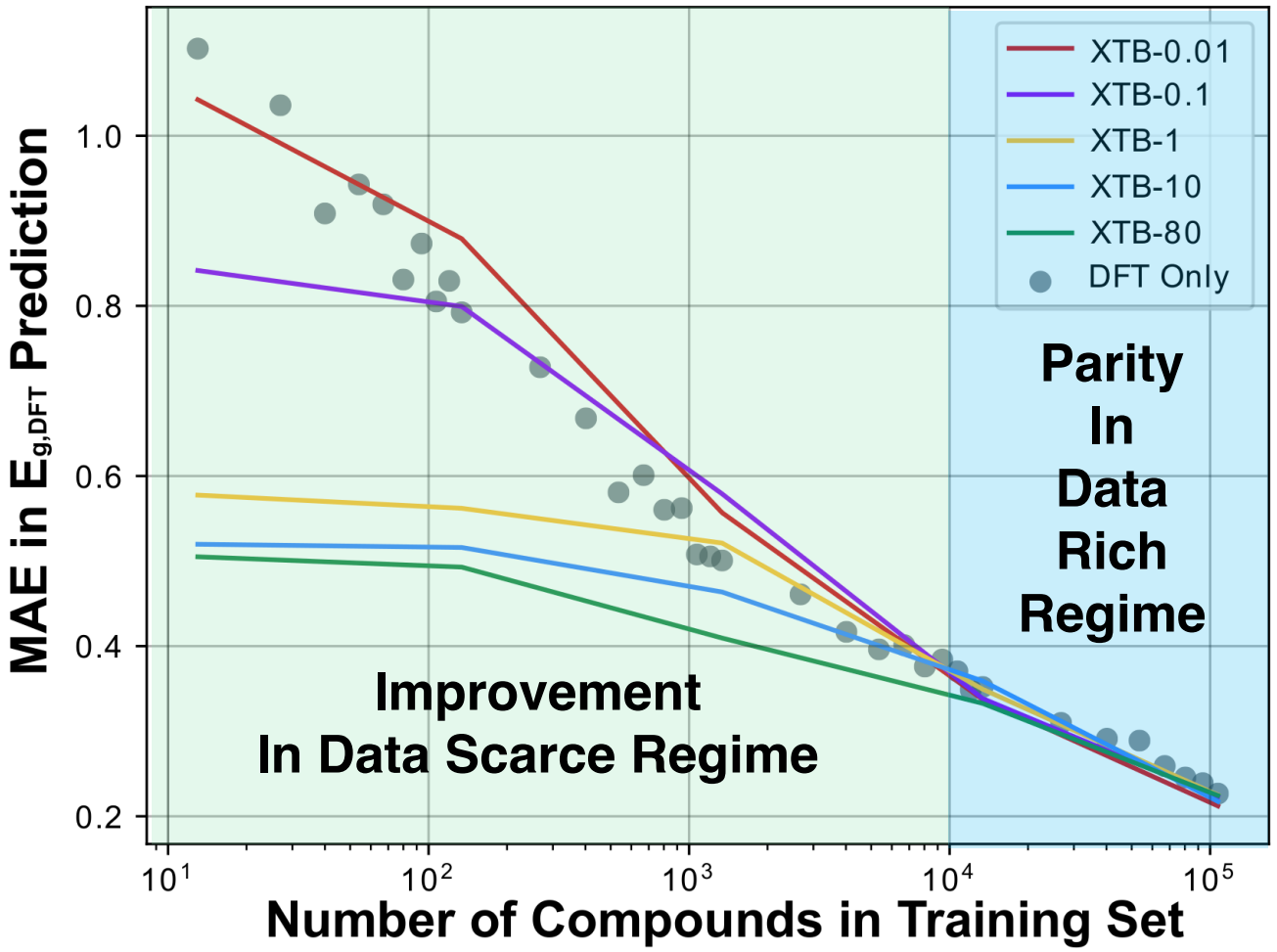
Transfer Learning on the Molecular Bandgap



Data Rich: XTB bandgap prediction (<1s per compound)

Data Scarce: DFT bandgap prediction (hours per compound)

Transfer Learning on the Molecular Bandgap

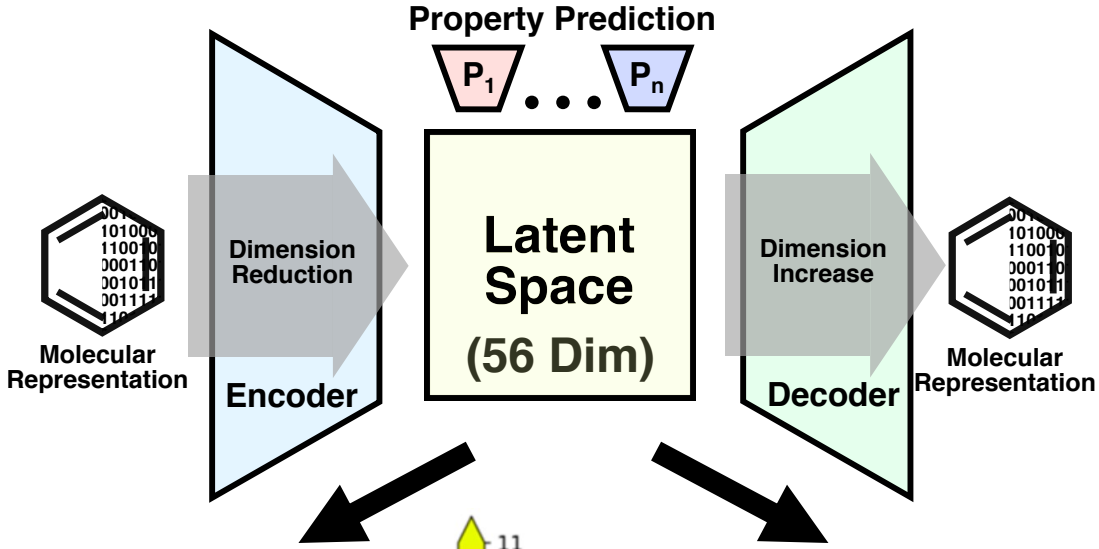


Data Rich: XTB bandgap prediction (<1s per compound)

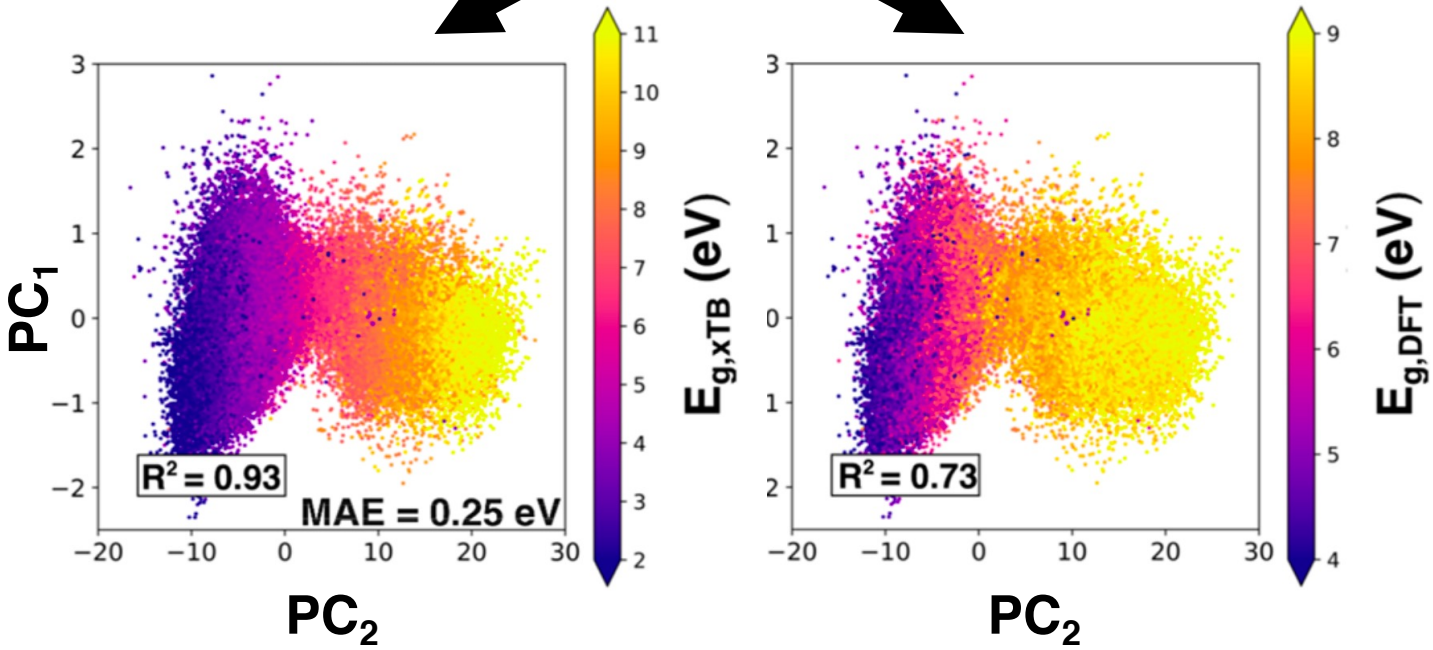
Data Scarce: DFT bandgap prediction (hours per compound)

The Effect of Property Prediction on Latent Space

What does including property prediction tasks mean for how the encoder behaves?

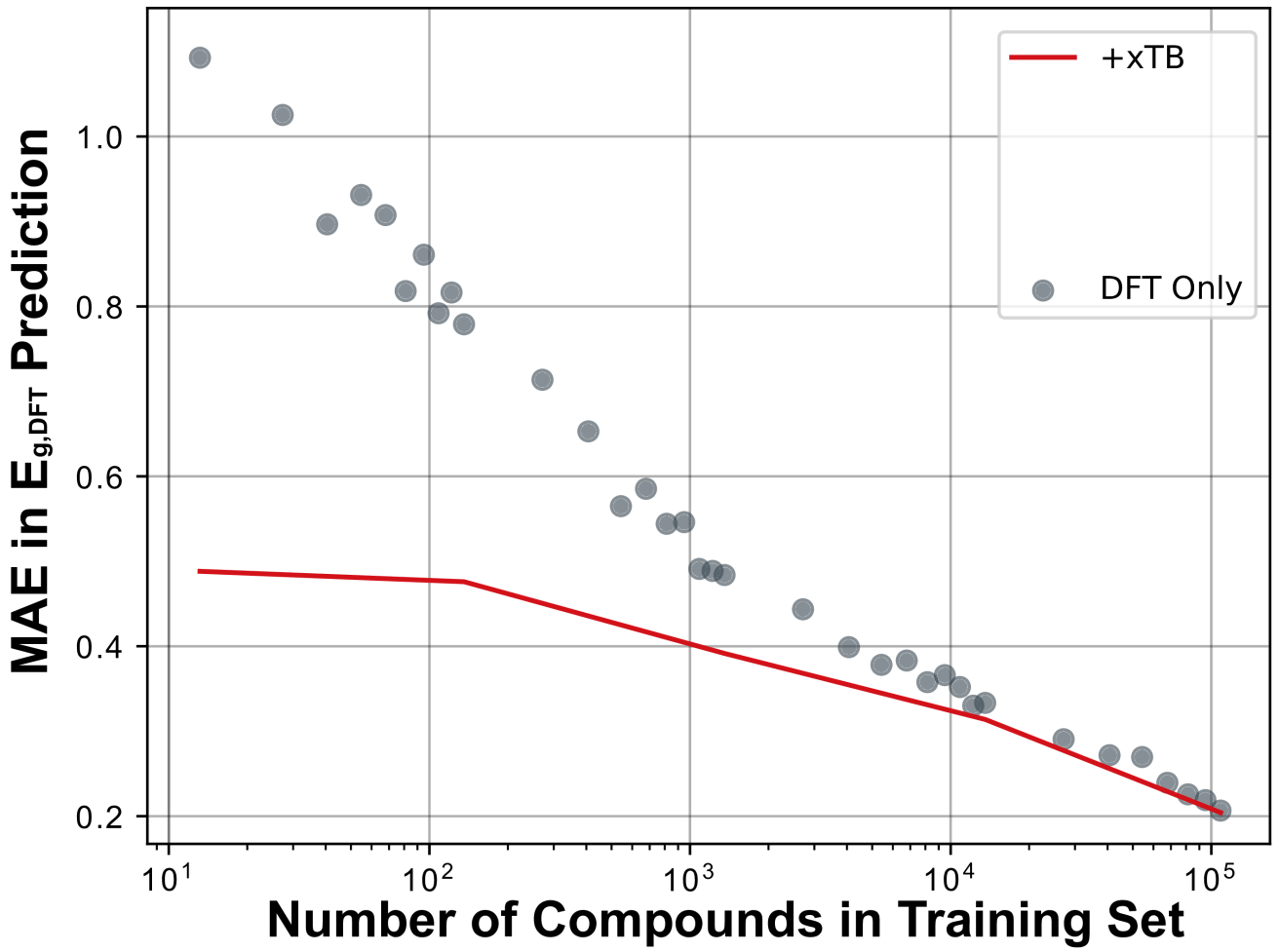


The encoder learns to arrange molecules with similar bandgaps to similar regions of the latent space



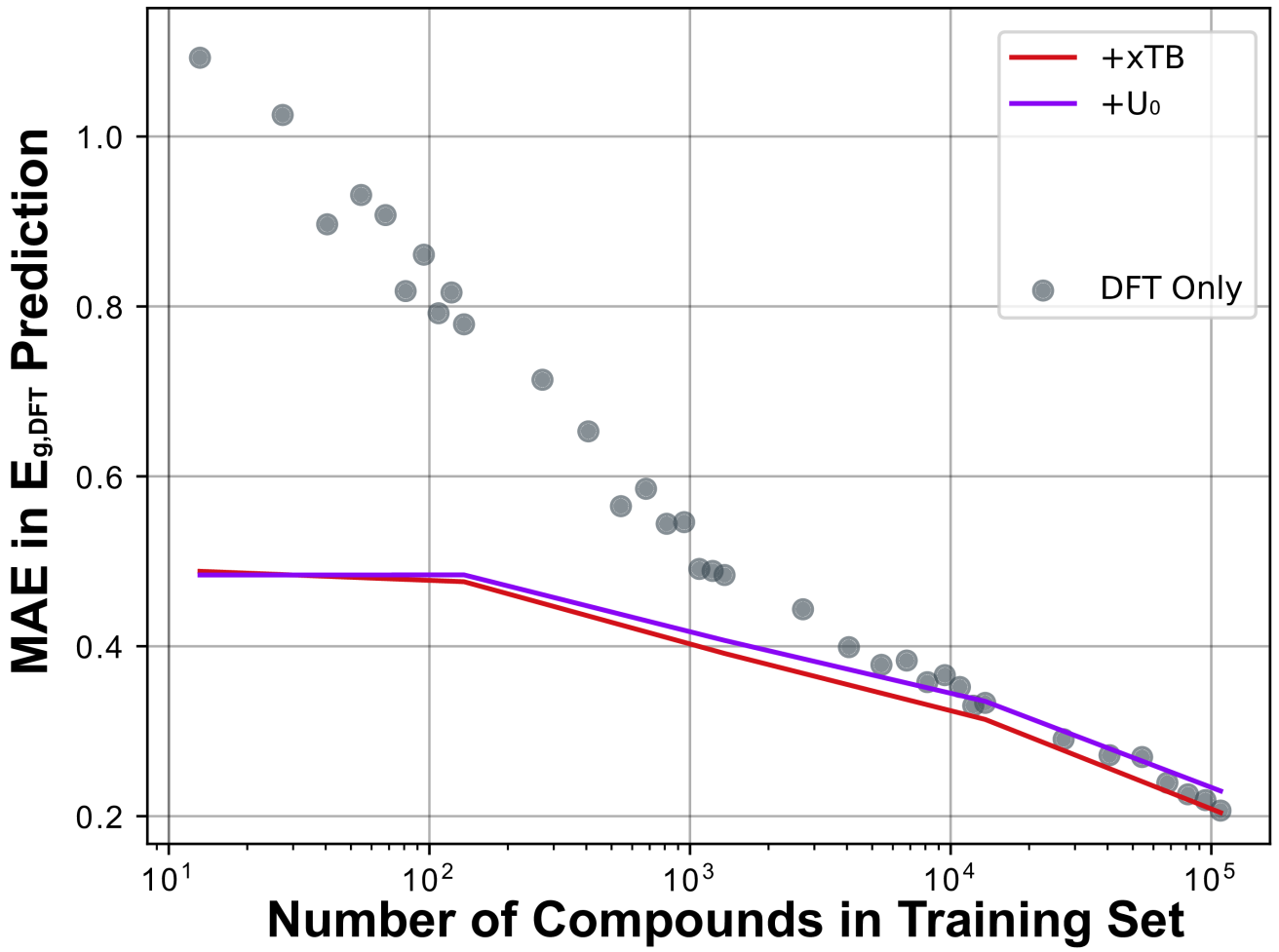
*Model trained only on E_{XTB} values, showing training + testing encodings

How Much More Information Can We Add?



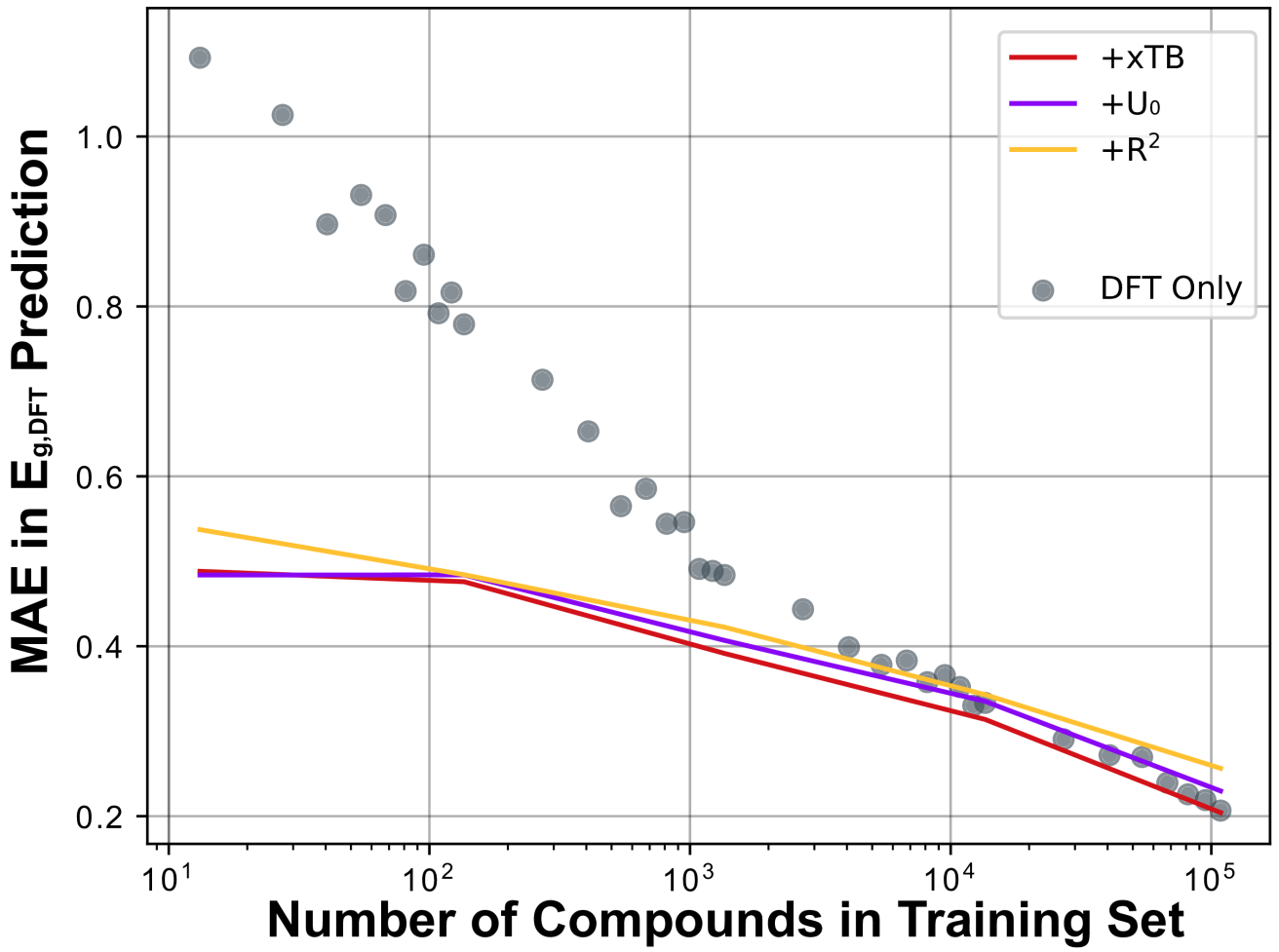
Enrichment with respect to other QM9 molecular properties (full 0.8 training fraction enrichment)

How Much More Information Can We Add?



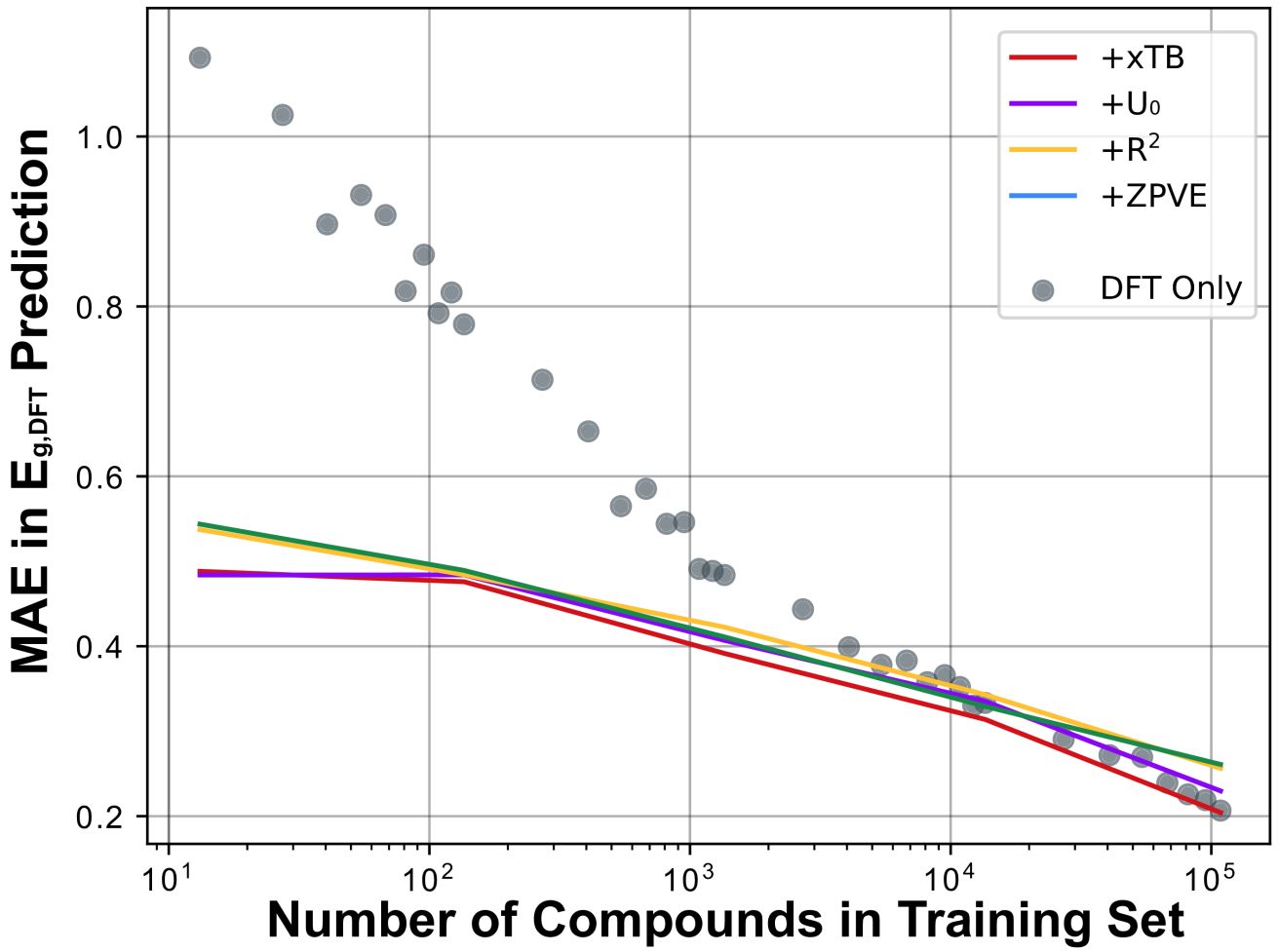
Enrichment with respect to other QM9 molecular properties (full 0.8 training fraction enrichment)

How Much More Information Can We Add?



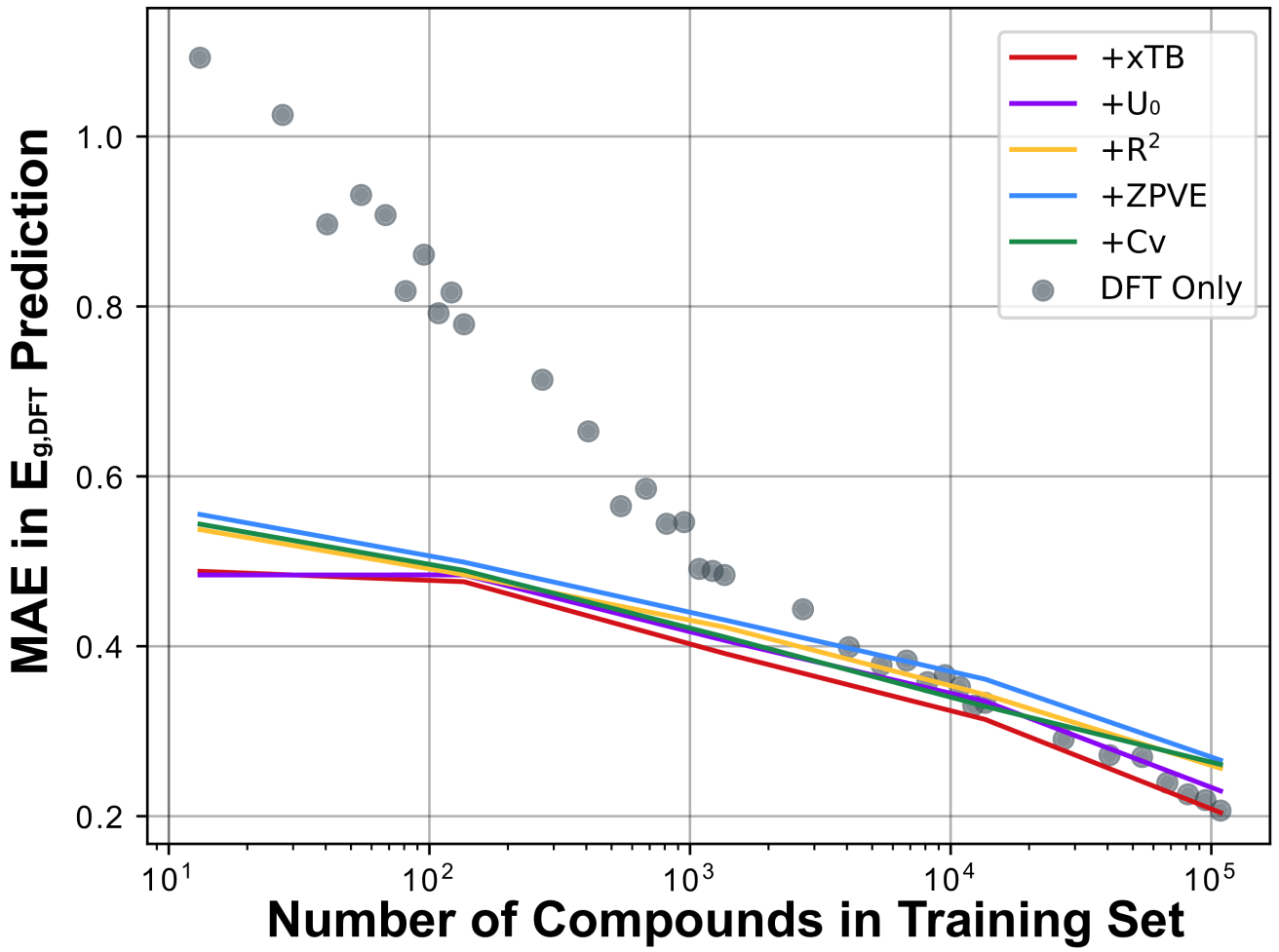
Enrichment with respect to other QM9 molecular properties (full 0.8 training fraction enrichment)

How Much More Information Can We Add?



Enrichment with respect to other QM9 molecular properties (full 0.8 training fraction enrichment)

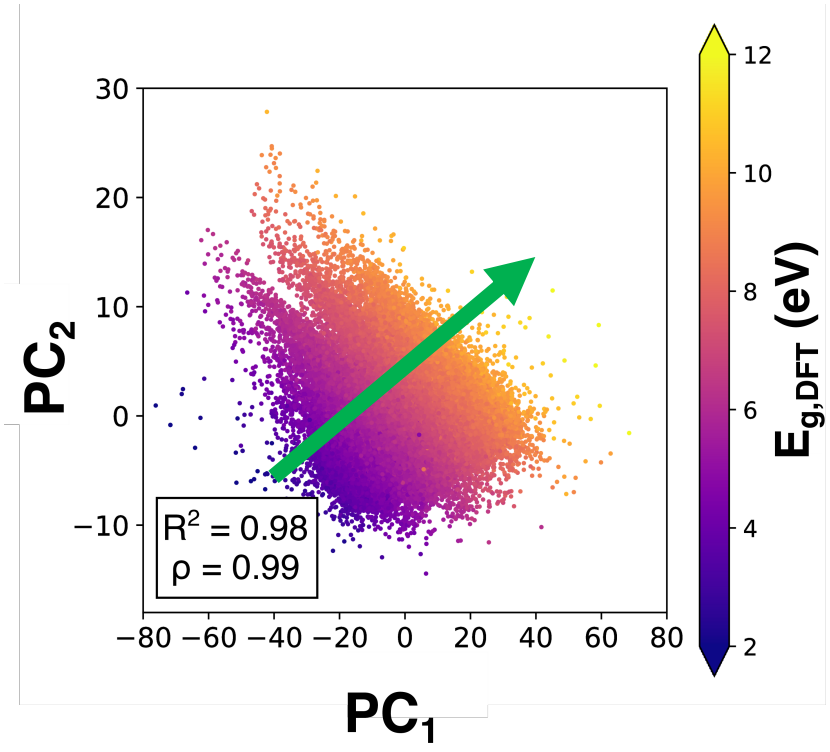
How Much More Information Can We Add?

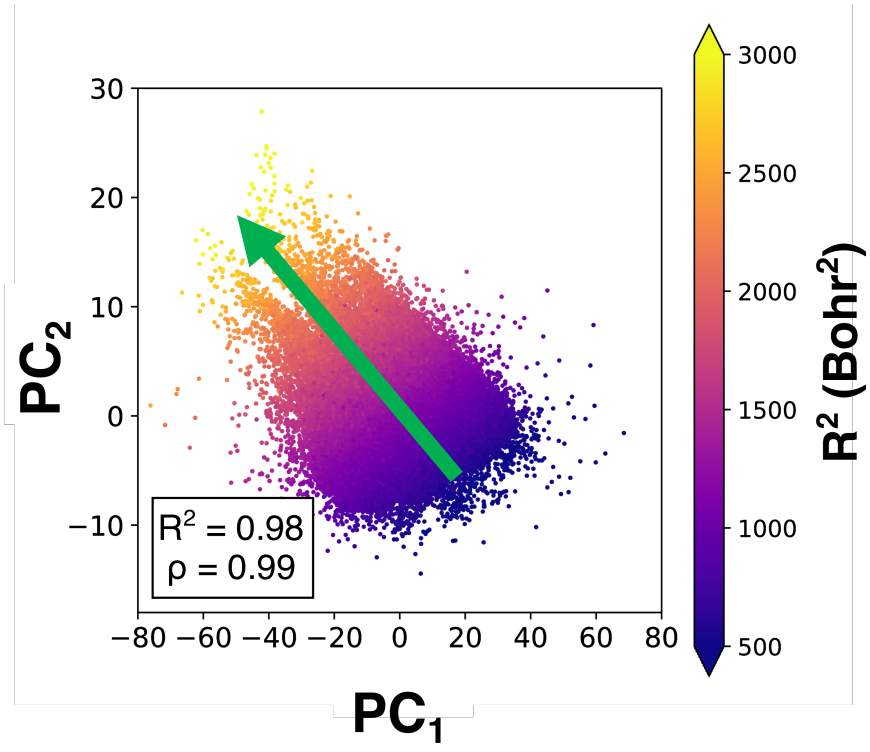


These additional properties are not correlated with bandgap, but they do add physical information and interpretability to the latent space.

Training on Multiple Properties Produces a More Physically Rich Latent Space

Latent Space Encodings for an E_g/R^2 Model

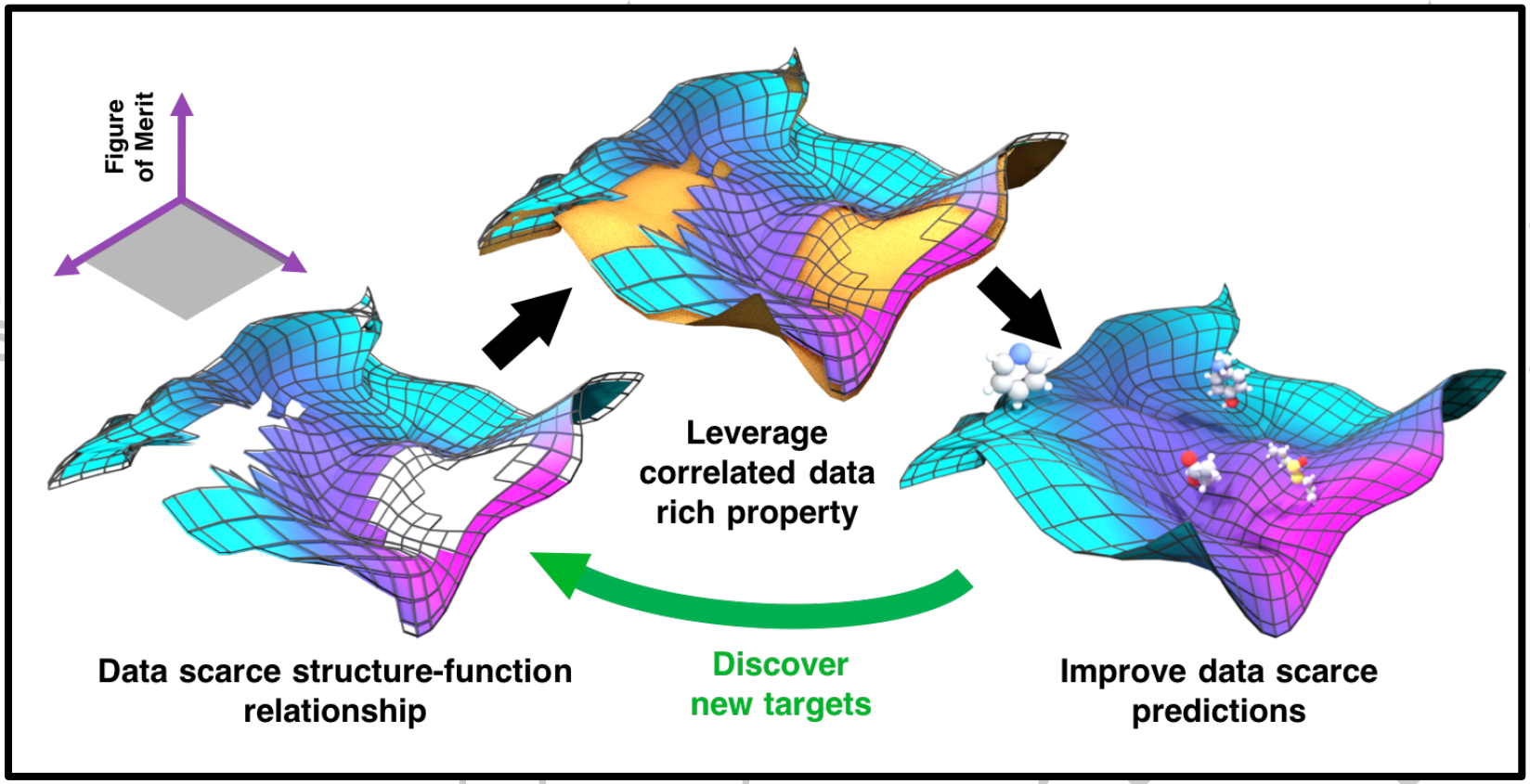




Non-correlated properties spontaneously organize along orthogonal latent space axes

Iovanac, N. C.; Savoie, B. M. "Simpler is Better: How Linear Prediction Tasks Improve Transfer Learning in Chemical Autoencoders." Submitted. **2020**

Training on Multiple Properties Produces a More Physically Rich Latent Space



Iovanac, N. C.; Savoie, B. M. "Simpler is Better: How Linear Prediction Tasks Improve Transfer Learning in Chemical Autoencoders." Submitted. **2020**

Does the Latent Space Contain Real Molecules Outside of the Training Chemistries?

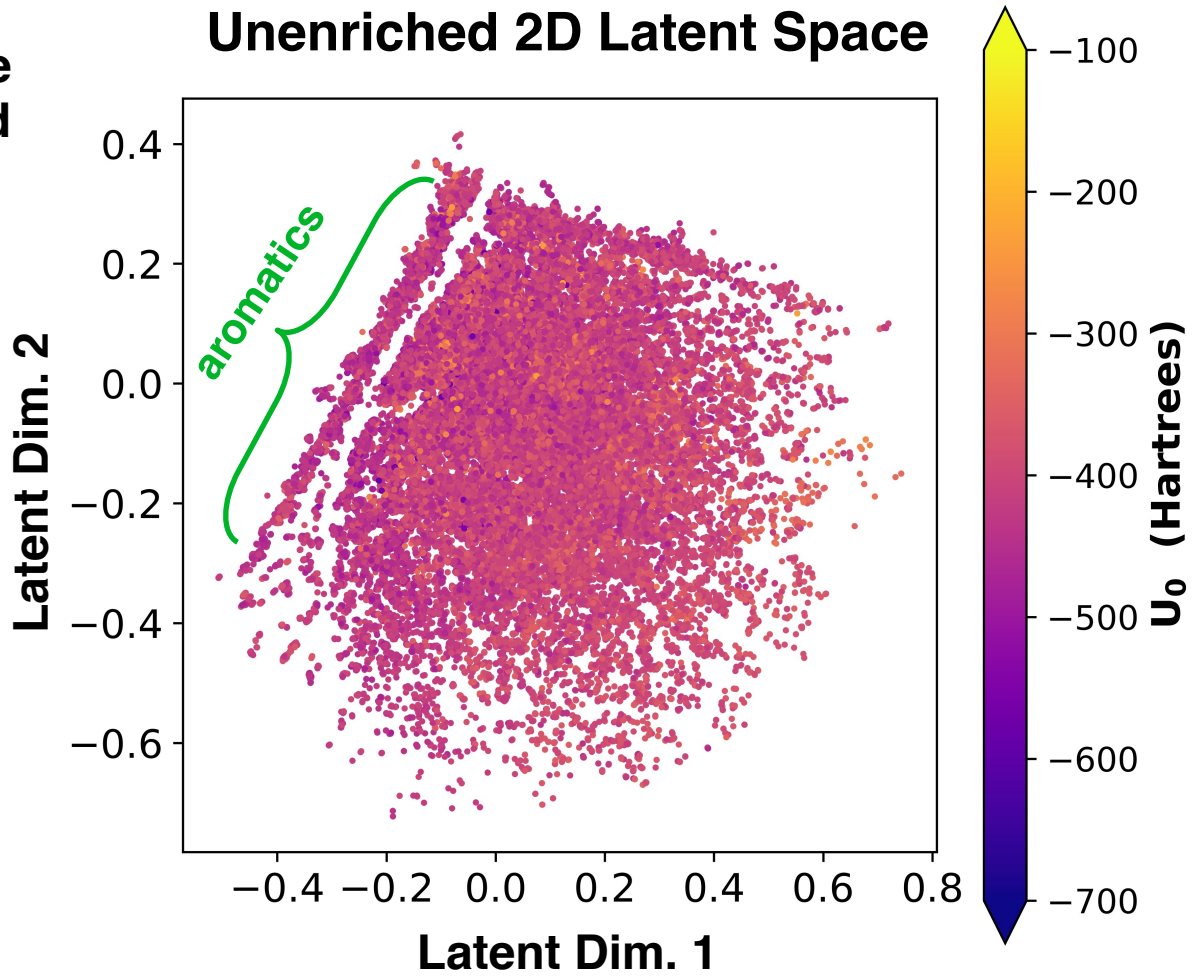
After training only ~4% of the latent space decodes to valid mappings

Gómez-Bombarelli, et al. *ACS Central Science* **2018**, *4* (2), 268–276.

This is partly due to SMILES being a fragile representation. With parse trees we already get a baseline validity of ~15%

Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar Variational Autoencoder. PMLR 70, 2017.

Note: 2D compression is too aggressive for accurate prediction



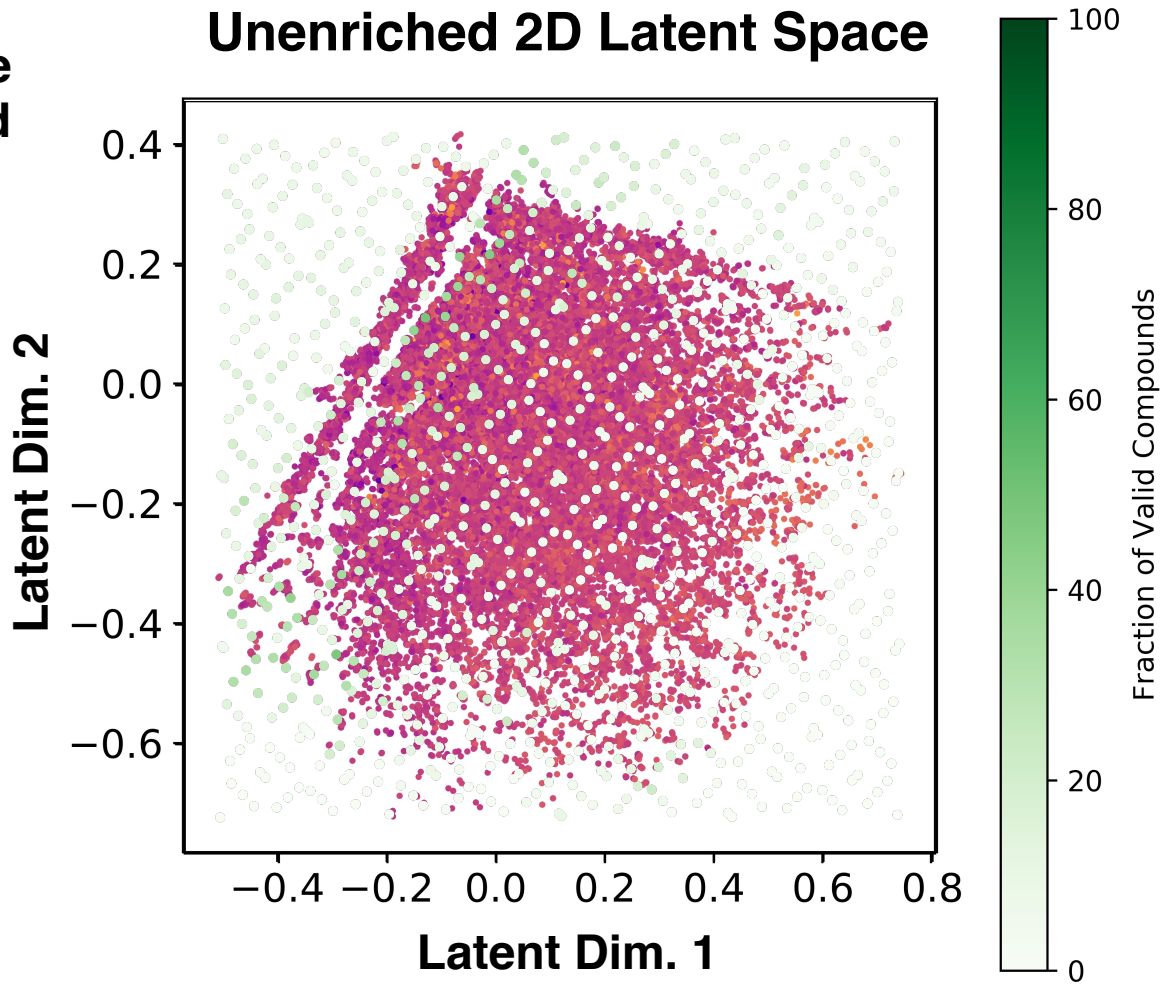
Does the Latent Space Contain Real Molecules Outside of the Training Chemistries?

After training only ~4% of the latent space decodes to valid mappings

Gómez-Bombarelli, et al. *ACS Central Science* **2018**, *4* (2), 268–276.

This is partly due to SMILES being a fragile representation. With parse trees we already get a baseline validity of ~15%

Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar Variational Autoencoder. PMLR 70, 2017.



Sobol (quasi-random space filling) sampling to interrogate validity

Baseline validity of ~10% over sampled points

Does the Latent Space Contain Real Molecules Outside of the Training Chemistries?

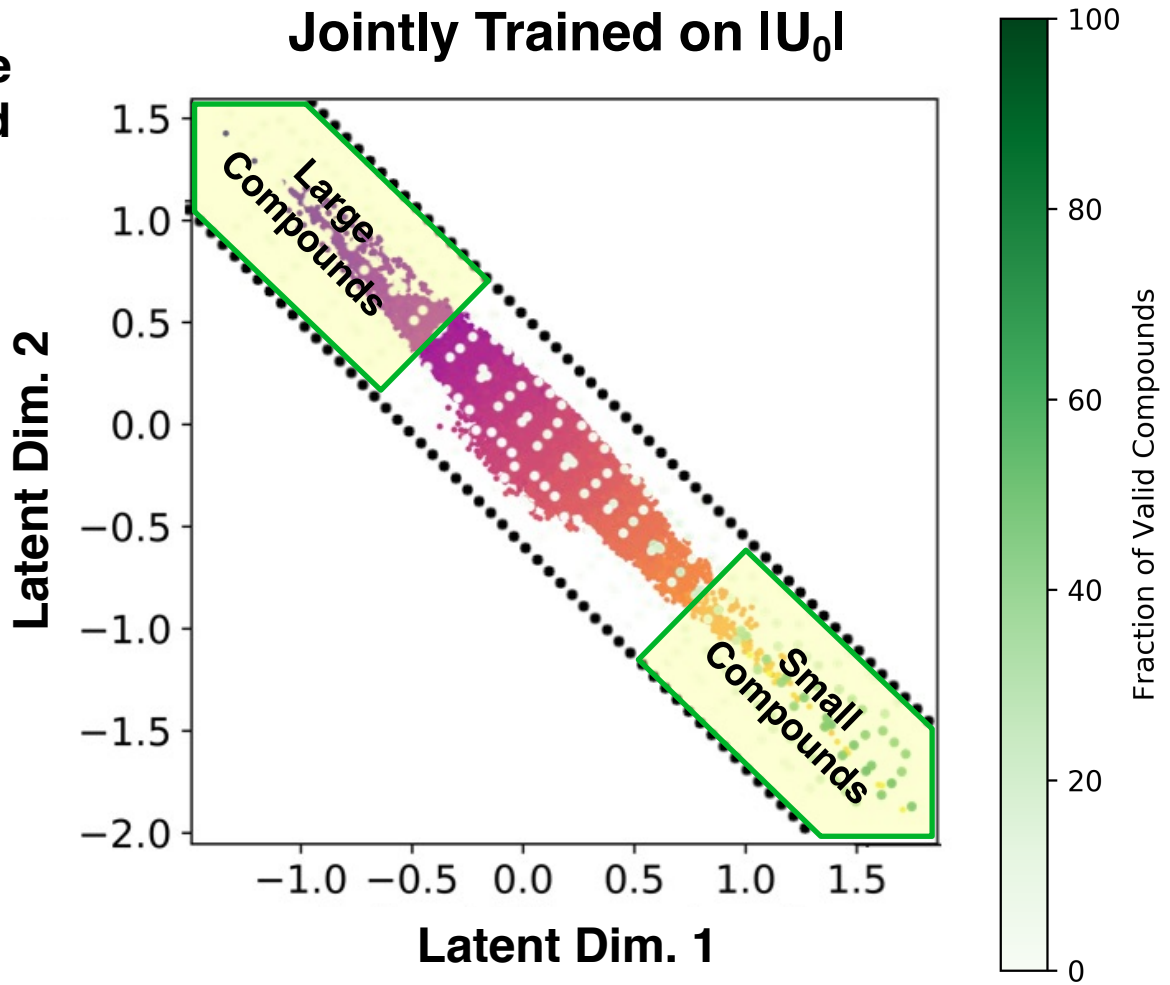
After training only ~4% of the latent space decodes to valid mappings

Gómez-Bombarelli, et al. *ACS Central Science* **2018**, *4* (2), 268–276.

This is partly due to SMILES being a fragile representation. With parse trees we already get a baseline validity of ~15%

Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar Variational Autoencoder. PMLR 70, 2017.

Validity trends become interpretable



High validity regions occur where we expect them and low validity in regions with a training data bias

Transfer Learning Improves Generative Capability

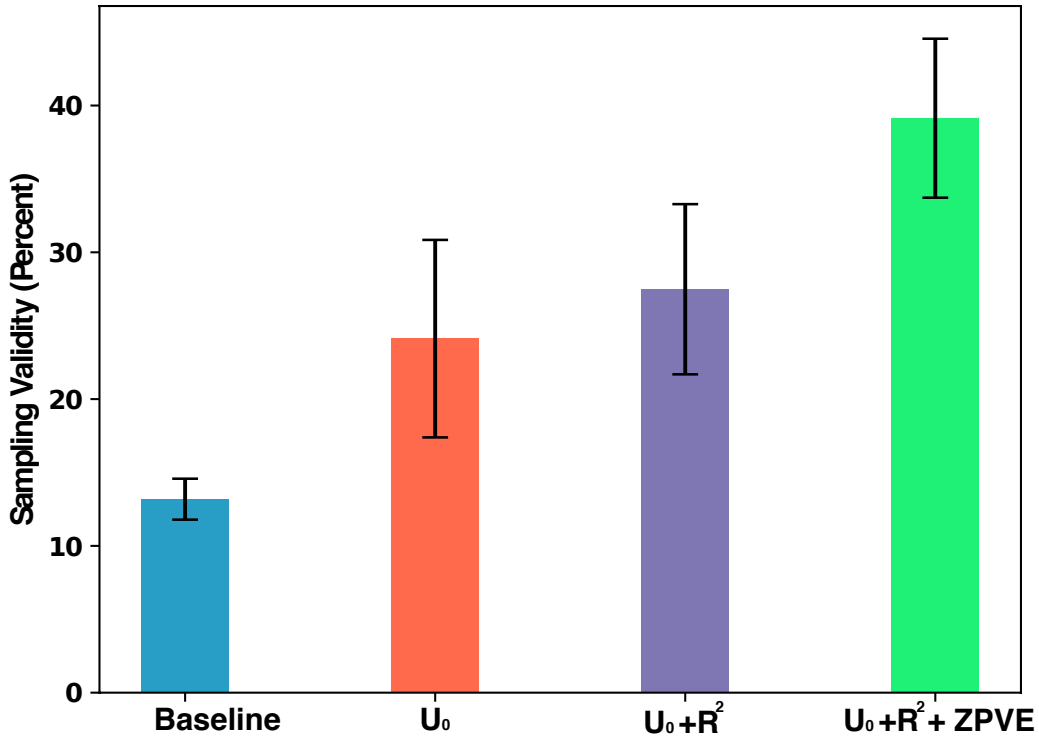
After training only ~4% of the latent space decodes to valid mappings

Gómez-Bombarelli, et al. *ACS Central Science* **2018**, *4* (2), 268–276.

This is partly due to SMILES being a fragile representation. With parse trees we already get a baseline validity of ~15%

Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar Variational Autoencoder. PMLR 70, 2017.

Validity for a 56-dimensional Enriched Latent Space

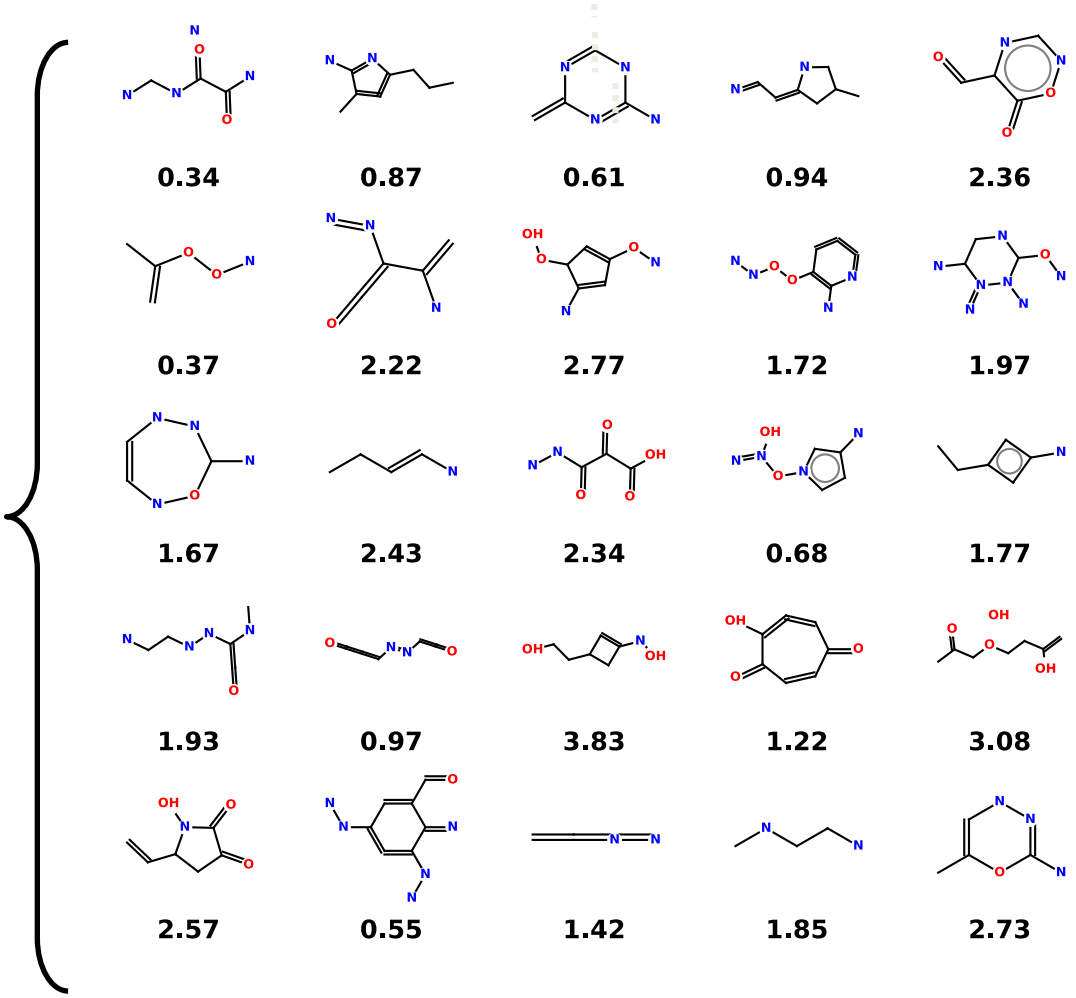


Multi-property training increases the interpretability and validity of structures found in the latent space

Transfer Learning Improves Generative Capability

New structures exhibit 100% diversity and 99% aren't even in QM9

Note: This is a hard task, because we are searching for small compounds with low bandgap; we end up with a lot of exotic looking structures



25/3000 randomly chosen low E_g structures

Outline

Transfer Learning:

Paradigms with examples:

Contemporary Topics:

Tutorial Example: COC(=O)CC(=C) ⟶ T_g, (E_{coh}, …)

Difference Learning Example

What we'll do:

- Ingest some data that I've scraped from the web related to the high MW limit T_g ~200 polymers.

- The dataset also includes the $T_{g,model}$ predictions of a semi-empirical model based on the cohesive energy of the monomer.

- Train a random forests model to predict $T_{g,exp}$ based on the repeat unit of the polymer (i.e., Morgan Fingerprint).

COC(=O)CC(=C) ⟶ $\mathbf{T_{g,exp}}$

- Train a random forests model to predict the difference $T_{g,exp} - T_{g,model}$

COC(=O)CC(=C) ⟶ $\mathbf{T_{g,exp} - T_{g,model}}$

Difference Learning Example

What we'll do:

- Ingest some data that I've scraped from the web related to the high MW limit T_g ~200 polymers.

- The dataset also includes the $T_{g,model}$ predictions of a se
m

- the repeat unit of the polymer (i.e., Morgan Fingerprint).

Caveat: T_g prediction is a hard problem, I've given you a very information rich $T_{g,model}$ variable which you would ordinarily have to work hard for.

COC(=O)CC(=C) ⟶ $T_{g,exp}$

- Train a random forests model to predict the difference $T_{g,exp} - T_{g,model}$

COC(=O)CC(=C) ⟶ $T_{g,exp} - T_{g,model}$

Example: Ingestion

Data Ingestion:

```
### Import polymer data
train_data = pd.read_csv("training.txt",sep='^')
test_data = pd.read_csv("testing.txt",sep='^')

### Import fingerprints (these are arrays so we use numpy directly)
train_prints = np.genfromtxt("training_FP.txt")
test_prints = np.genfromtxt('testing_FP.txt')

### Print diagnostics
print("Number of samples in training set: {}".format(len(train_data["Name"])))
print("Number of samples in testing set: {}".format(len(test_data["Name"])))
print("\nPolymer data columns and dtypes:")
for _ in train_data.columns: print("{}: {}".format(_,train_data[_].dtype))
print("Example of a Morgan Fingerprint: {}".format(train_prints[0]))
```

```
Number of samples in training set: 195
Number of samples in testing set: 20
Polymer data columns and dtypes:
Name: object
SMILES: object
TG_ref: float64
TG_model: float64
Ecoh: float64
Example of a Morgan Fingerprint: [1. 0. 1. ... 0. 0. 0.]
```

Example: Exploring the Data

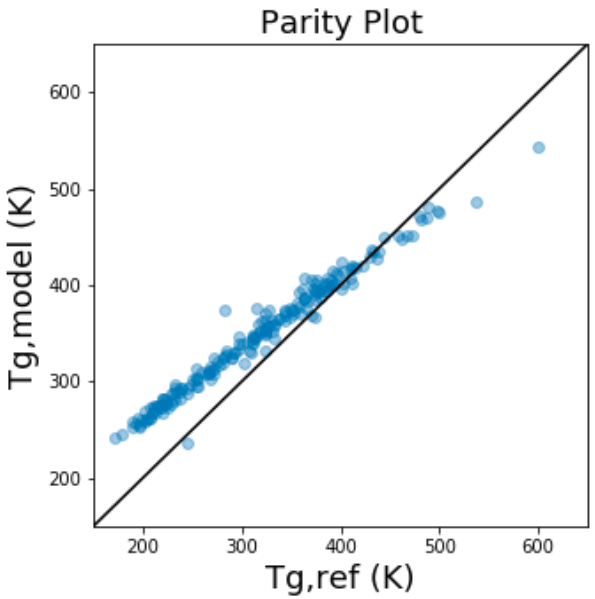
Overview of Supplied Data:

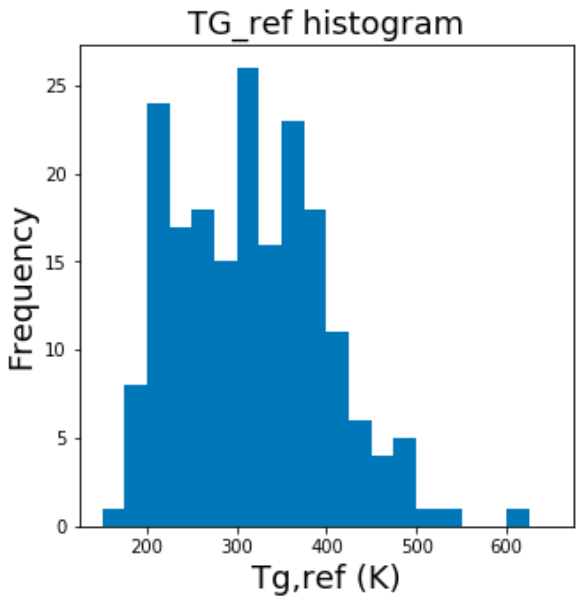
```
# Compute and print summary statistics for TG_model compared with TG_ref
ae = np.absolute(test_data["TG_model"]-test_data["TG_ref"])
print('Mean |Tg-Tg,model| (AE): {:< 4.2f}'.format(np.mean(ae)))
print('Mean %AE:  {:< 4.2f}'.format(np.mean(ae/test_data["TG_ref"]*100.0)))
print('StdAE:     {:< 4.2f}'.format(np.std(ae)))
print('Median AE: {:< 4.2f}'.format(np.median(ae)))
```

```
Mean |Tg-Tg,model| (AE):    33.63
Mean %AE:   12.77
StdAE:      19.54
Median AE:  30.43
```

$T_{g,model}$ exhibits a systematic bias with a slight non-linearity

More typically we would be dealing with a more complicated relationship between the transfer and target variables





Example: Single-Task Model

• As a baseline we will train random forest model on the `fingerprint` → `TG_ref` prediction task.

• I have already done a crude optimization over the hyperparameters `max_depth` (10,20,30,100,1000) and `n_estimators` (trees) (10,100,1000) using 10-fold cross-validation on the training set to determine the best average performance across all splits. **(Never use your test data until evaluation)**

Train the model:

```
# Initialize and train a random forest model to predict Tg_ref directly
# Note: setting the random seed for reproducibility
single_model = RandomForestRegressor(max_depth=100,random_state=0,n_estimators=1000)
single_model.fit(train_prints,train_data["TG_ref"])
```

The training/testing split in this case is 90:10 and our x-feature is the Morgan fingerprint of each polymer's repeat unit.

Example: Single-Task Model

Evaluate Single-Task Model Performance:

```
# Calculate the Tg predictions with the trained model
test_p = np.squeeze(single_model.predict(test_prints))
train_p = np.squeeze(single_model.predict(train_prints))

# Compute and print summary statistics
ae = np.absolute(test_p-test_data["TG_ref"])
print('Summary test statistics for Tg,ref single task model:\n')
print('Mean AE:   {:< 4.2f}'.format(np.mean(ae)))
print('Mean %AE:  {:< 4.2f}'.format(np.mean(ae/test_data["TG_ref"]*100.0)))
print('StdAE:     {:< 4.2f}'.format(np.std(ae)))
print('Median AE: {:< 4.2f}'.format(np.median(ae)))
```

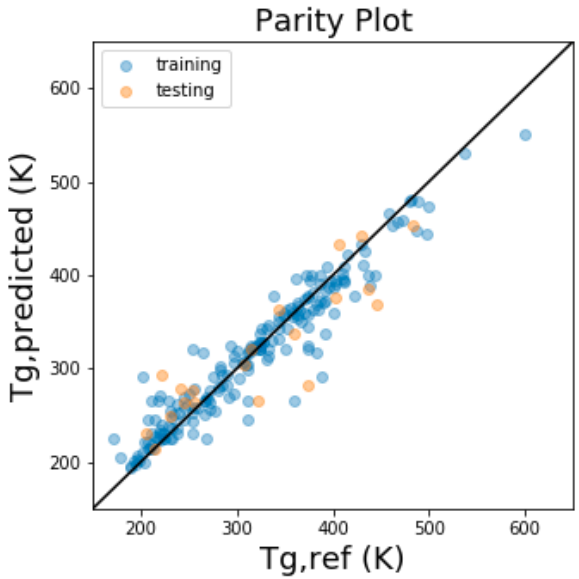
```
Mean AE:    31.34
Mean %AE:   9.84
StdAE:      24.87
Median AE:  24.33
```

Example: Single-Task Model

Evaluate Single-Task Model Performance:

```
# Parity plot of ref vs predicted Tg in training data and testing data
plt.figure(figsize=(5,5))
plt.plot([150,650],[150,650],color="k")
plt.scatter(train_data["TG_ref"],train_p,alpha=0.4,label="training")
plt.scatter(test_data["TG_ref"],test_p,alpha=0.4,label="testing")
plt.xlim(150,650)
plt.ylim(150,650)
plt.xlabel("Tg,ref (K)",size=18)
plt.ylabel("Tg,predicted (K)",size=18)
plt.title("Parity Plot",size=18)
plt.legend()
plt.show()
```

All things considered, the model doesn't perform too poorly. T_g is a hard property to learn, and we've given a relatively generic feature (i.e., a fingerprint) to learn from.



Example: Difference Model

As our transfer learning example we will train a random forest model to predict the difference between `TG_ref` and `TG_model`.

The idea is that we have a model that already captures a lot of the physics of T_g and is cheap to calculate (i.e., we can also calculate it for any new polymers that aren't in the training set). Learning the difference between the real T_g and the model T_g should be easier than trying to learn the real T_g from scratch.

Train the difference model:

```
# Train a random forest model to predict the Tg difference (Tg_ref-Tg_model)
# Note: setting the random seed for reproducibility
diff_model = RandomForestRegressor(max_depth=10,random_state=0,n_estimators=10)
diff_model.fit(train_prints,train_data["TG_ref"]-train_data["TG_model"])
```

Again, I have already performed a hyperparameter optimization outside of the notebook. Here we have just trained the model with the hyperparameters I optimized from cross-validation on the training data.

Example: Difference Model

Evaluate the Difference Model Performance:

```
# Calculate the difference predictions with the trained model
# NOTE: Actual Tg_ref predictions are the difference plus TG_model values
test_dp = np.squeeze(diff_model.predict(test_prints))
test_p = test_dp + test_data["TG_model"]
train_dp = np.squeeze(diff_model.predict(train_prints))
train_p = train_dp + train_data["TG_model"]

# Compute and print summary statistics
ae = np.absolute(test_p-test_data["TG_ref"])
print('Summary test statistics for difference model:\n')
print('Mean AE:   {:< 4.2f}'.format(np.mean(ae)))
print('Mean %AE:  {:< 4.2f}'.format(np.mean(ae/test_data["TG_ref"]*100.0)))
print('StdAE:     {:< 4.2f}'.format(np.std(ae)))
print('Median AE: {:< 4.2f}'.format(np.median(ae)))
```

```
Mean AE:    14.56
Mean %AE:   4.54
StdAE:      8.73
Median AE:  14.91
```

Note: to evaluate the T_g predictions we need to add the `Tg_model` values for the test set to the differences.

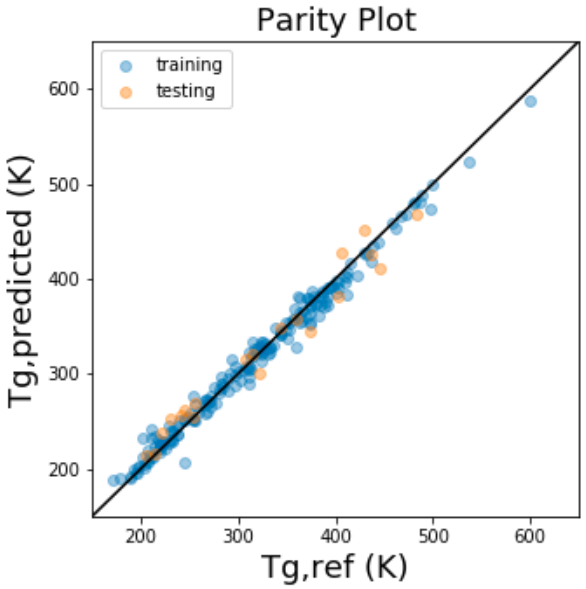
Example: Difference Model

Evaluate the Difference Model Performance:

```
# Parity plot of ref vs predicted Tg in training data and testing data
plt.figure(figsize=(5,5))
plt.plot([150,650],[150,650],color="k")
plt.scatter(train_data["TG_ref"],train_p,alpha=0.4,label="training")
plt.scatter(test_data["TG_ref"],test_p,alpha=0.4,label="testing")
plt.xlim(150,650)
plt.ylim(150,650)
plt.xlabel("Tg,ref (K)",size=18)
plt.ylabel("Tg,predicted (K)",size=18)
plt.title("Parity Plot",size=18)
plt.legend()
plt.show()
```

The difference model performs strikingly better than the single-task model

By only focusing on learning the difference, we have transferred information from our empirical model into the original learning task



Example: Difference Model

Evaluate the Difference Model Performance:

```
# Parity plot of ref vs predicted Tg in training data and testing data
plt.figure(figsize=(5,5))
plt.plot([150,650],[150,650],color="k")
plt.scatter(train_data["TG_ref"],train_p,alpha=0.4,label="training")
plt.scatter(test_data["TG_ref"],test_p,alpha=0.4,label="testing")
plt.xlim(150,650)
plt.ylim(150,650)
plt.x
plt.y
plt.t
plt.l
plt.s
```

I'll stress again that this is an artificial example, where regressing the model values would have provided similar performance, but the concept generalizes to more complicated relationships between TL variables and target tasks.

The
bett

By only focusing on learning the difference, we have transferred information from our empirical model into the original learning task