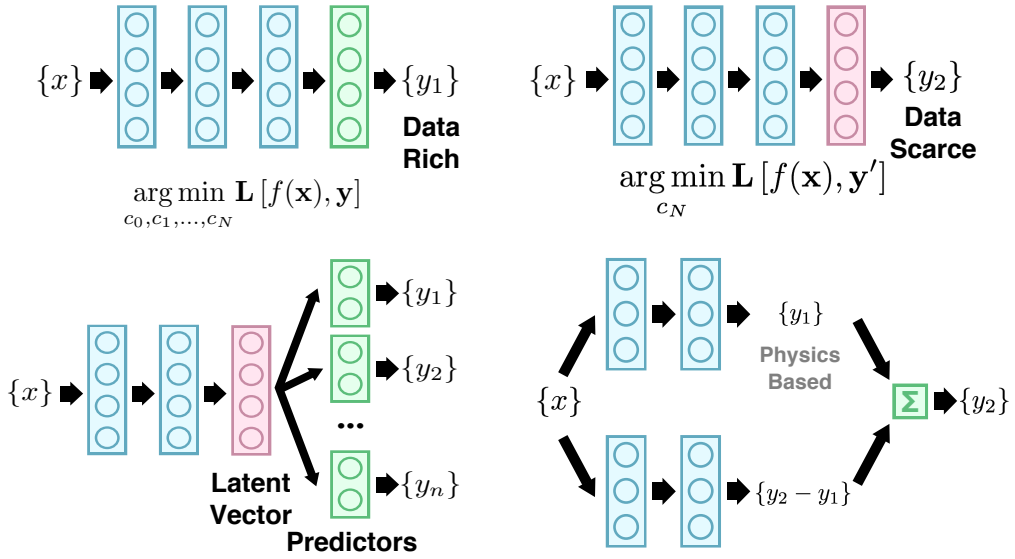
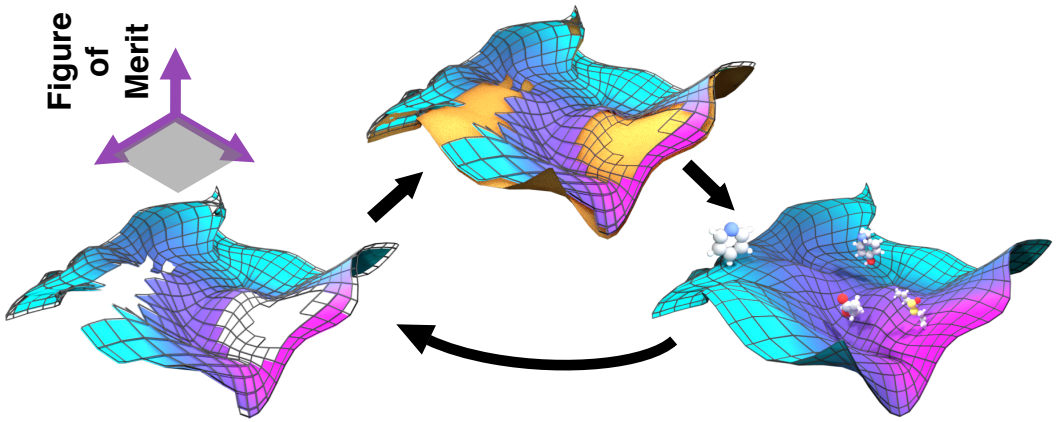
Goals for Today:

Paradigms with examples:



Contemporary Topics:



Tutorial Example:

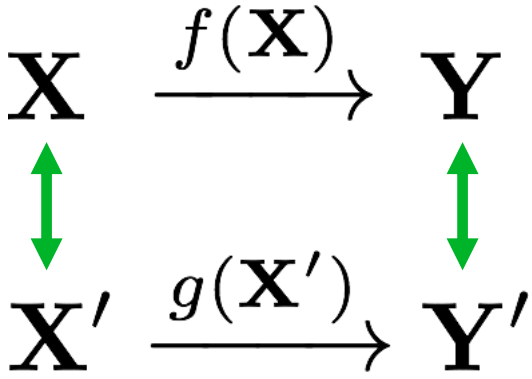
COC(=O)CC(=C) ⟶ T_g, (E_{coh}, …)

What is Transfer Learning?

Consider two supervised learning problems, where $n > N$:

$$\mathbf{D} : \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)\}$$

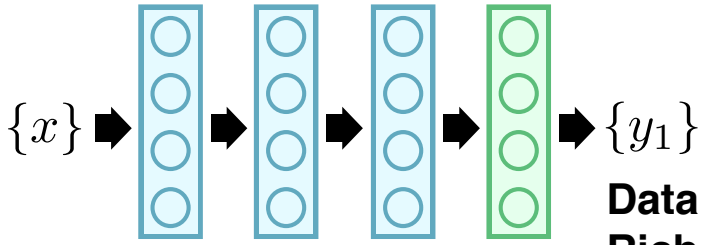
$$\mathbf{D}' : \{(\mathbf{x}'_1, \mathbf{y}'_1), ..., (\mathbf{x}'_N, \mathbf{y}'_N)\}$$

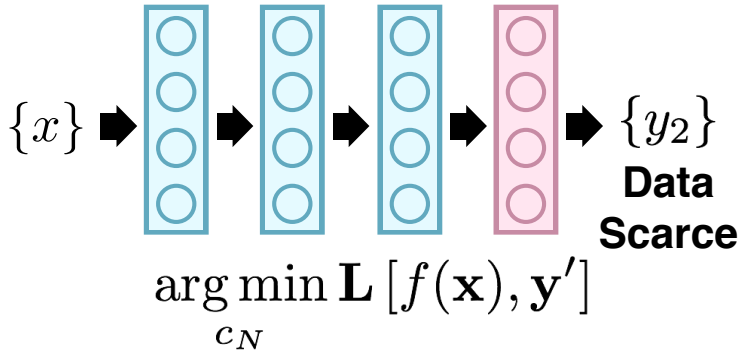


Transfer learning aims to improve the learning of $g(\mathbf{X}')$
using information from $\mathbf{D}$ *or* $f(\mathbf{X})$,
where $\mathbf{D}' \neq \mathbf{D}$ *and* $(f(\cdot), \mathbf{Y}) \neq (g(\cdot), \mathbf{Y}')$

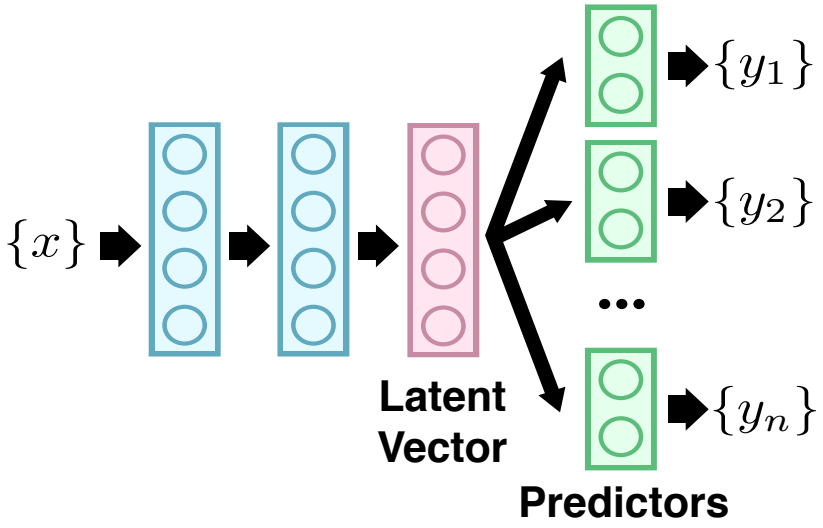
Transfer Learning – Common Paradigms

Transfer through Retraining

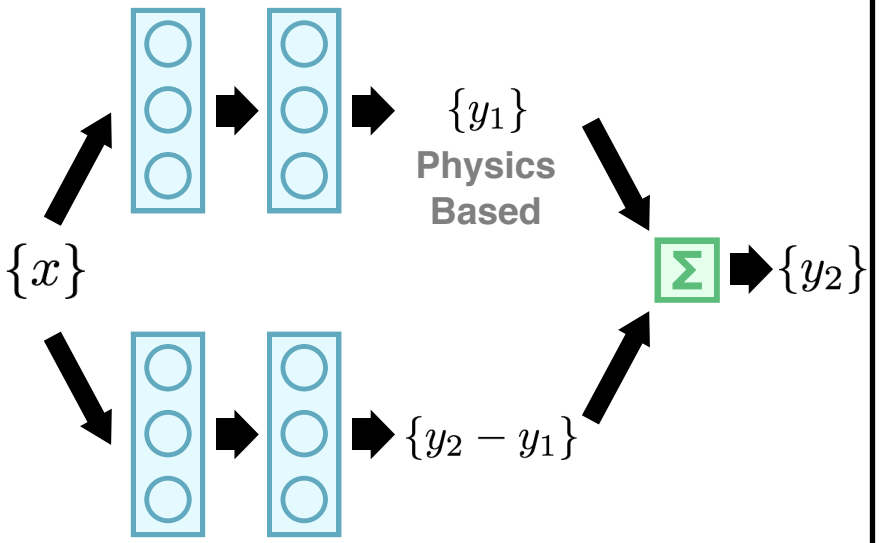




Multitask (Latent) Transfer



Difference Transfer



Transfer Learning – Basic Mechanisms

Regularization

TL can exhibit prediction improvements even in scenarios where the tasks/datasets aren't logically related.

Including more information tends to regularize model predictions (similar to L1/L2/Elastic Net) especially in extrapolative scenarios.

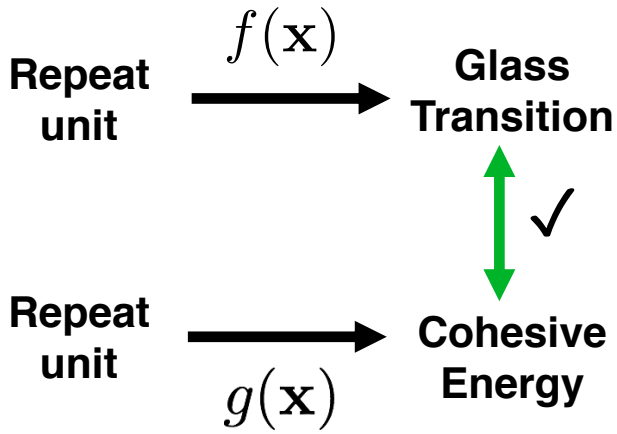
$$\underset{c_0,c_1,\ldots,c_N}{\arg\min}\ \mathbf{L}\left[f(\mathbf{x}),\mathbf{y}\right]$$

vs

$$\underset{c_0,c_1,\ldots,c_N}{\arg\min}\ \mathbf{L}\left[f(\mathbf{x}),\mathbf{y},\mathbf{y}',\mathbf{y}'',\ldots\right]$$

Task Correlation

TL can leverage information from one learning task for a related task when they share information (not necessarily linear).



*you'll also see this and the next category described as types of "inductive" transfer

Pan, S. J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knol. Data Eng.* **2010**, *22* (10), 1345–1359.

Weiss, K.; Khoshgoftaar, T. M.; Wang, D. A Survey of Transfer Learning. *J. Big Data* **2016**, *3* (1).

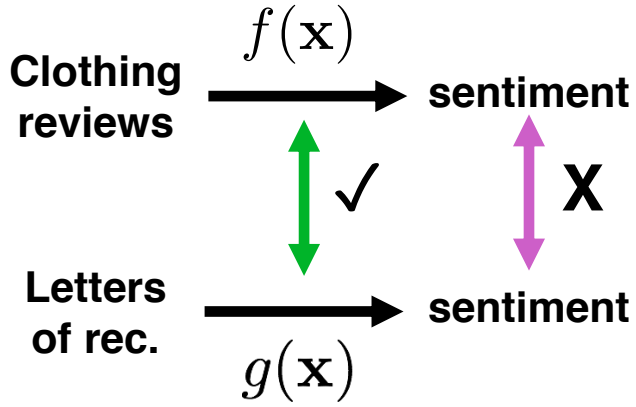
Hutchinson, M. L.; Antono, E.; Gibbons, B. M.; Paradiso, S.; Ling, J.; Meredig, B. Overcoming Data Scarcity with Transfer Learning. arXiv, 2017.

Transfer Learning – Basic Mechanisms

Common Latent Representation

If predictors are unrelated but conditioned on the same features, information exchange can occur.

This is distinct from correlation of prediction properties, since the information exchange occurs through a better internal representation (latent) of the feature distribution.

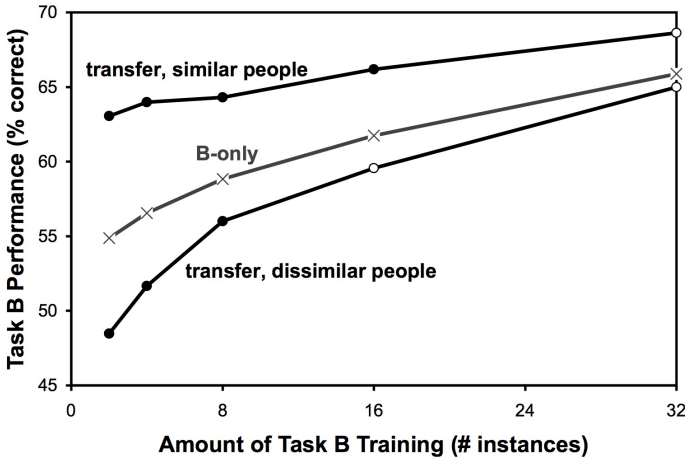


*you'll also see this and the previous category described as types of "inductive" transfer

Negative Transfer

TL can lead to negative results when the data rich task is not representative of the data scarce domain (e.g., contains a bias that is inherited by the data scarce task).

Particularly prevalent when there are large data imbalances and sensitive to the model architecture (contrasts with regularization behavior).



Rosenstein, M. T.; Marx, Z.; Kaelbling, L. P.; Dietterich, T. G. NIPS 05 Workshop Paper

Transfer Learning – Basic Mechanisms

Common Latent Representation

If predictors are unrelated but conditioned on the same features, information exchange can occur.

This is distinct from correlation of prediction prop… throu… the f…

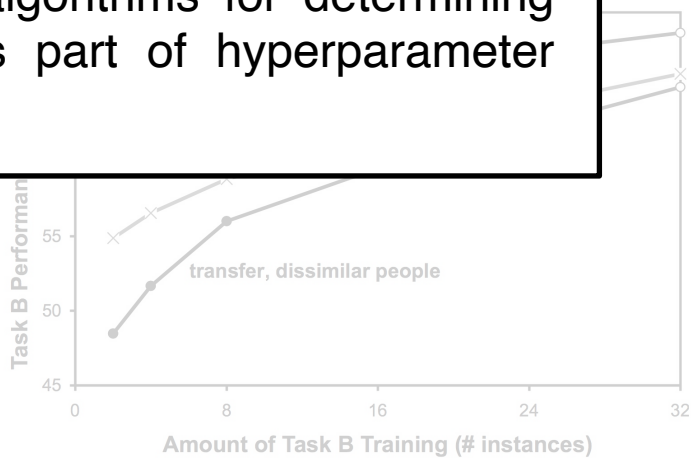
Example: Let's say you have abundant data on vernacular English usage from Twitter. Now, you want to train a model to translate classic texts, would the twitter data be useful?

It depends… There are a lot of heuristic algorithms for determining optimal TL policies, but experimentation as part of hyperparameter optimization is still standard.

Ne…

TL c… task is not representative of the data scarce domain (e.g., contains a bias that is inherited by the data scarce task).

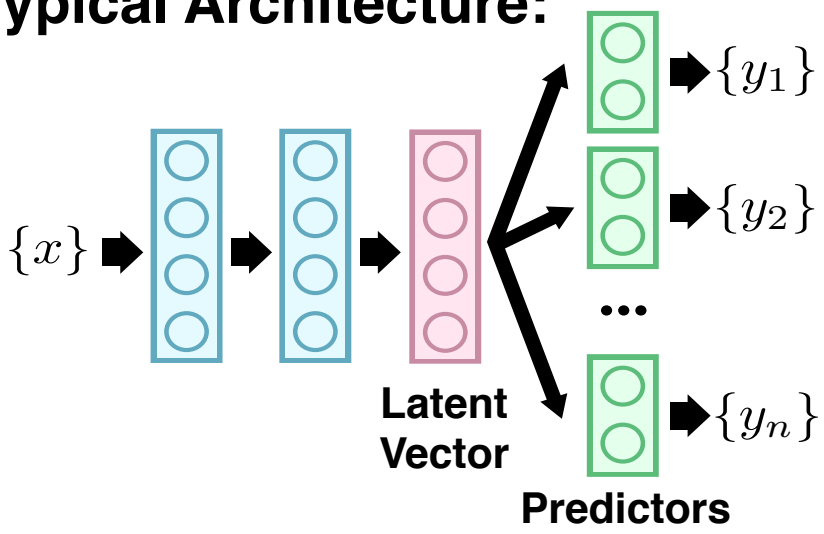
Particularly prevalent when there are large data imbalances and sensitive to the model architecture (contrasts with regularization behavior).



Rosenstein, M. T.; Marx, Z.; Kaelbling, L. P.; Dietterich, T. G. NIPS 05 Workshop Paper

Multitask Learning - Concept

Typical Architecture:



Concept:

x is a common feature for tasks on $y_1, y_2, \ldots, y_n$

$y_1 \ldots y_n$ may not be obviously related, but given that they are conditioned on **x**, $f(x) \rightarrow y_1$ may contain relevant information for $f(x) \rightarrow y_2$

Terminology Note: in the literature some people only consider multitask a TL strategy if you are only trying to improve y_T (and not the other y_i).

When you would use it:

1) You have data for many properties of the same compound/material

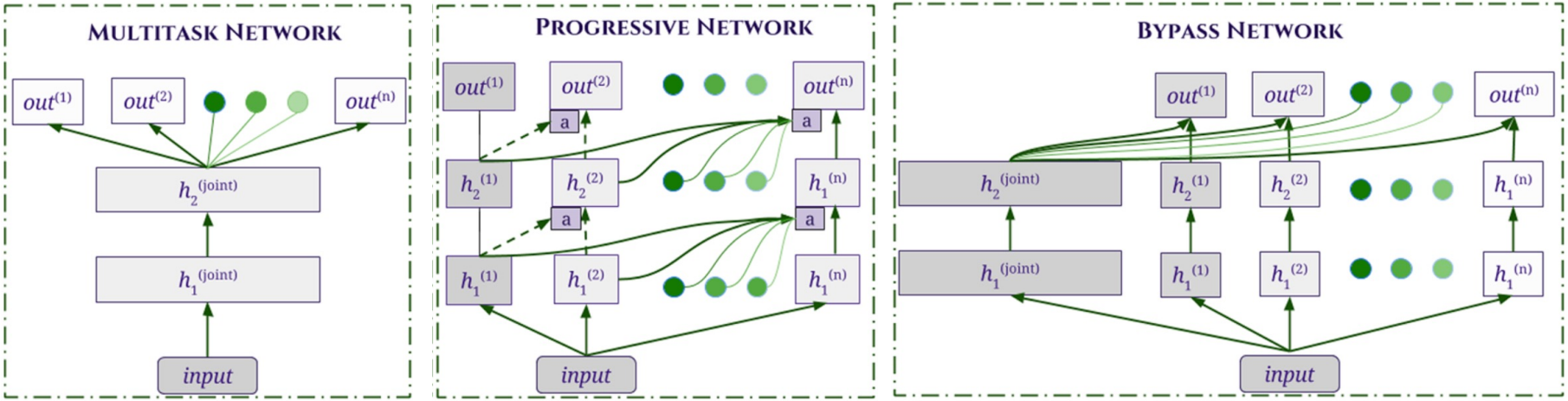
2) You have a large data disparity. E.g., lots of data for properties y_i but only a small amount of data for y_T.

3) You need to regularize your model. E.g., you are using a complex predictor and overfitting is a concern.

Multitask Learning - Example

Ramsundar, B.; Liu, B.; Wu, Z.; Verras, A.; Tudor, M.; Sheridan, R. P.; Pande, V. Is Multitask Deep Learning Practical for Pharma? *J. Chem. Info. Model.* **2017**, *57* (8), 2068–2076.

Compare three flavors of multitask:



Multitask: All prediction tasks share a common set of weights.

Progressive: Prediction tasks are trained individually, with dedicated weights and feedback via task-specific weights

Bypass: Hybrid approach, where tasks share a common set of weights (joint) and a dedicated column for each task

Multitask Learning - Example

- Investigate multitask architectures on several datasets (Kaggle, Protease inhibitors, Kinase inhibitors) compared with random forests and single task NN.

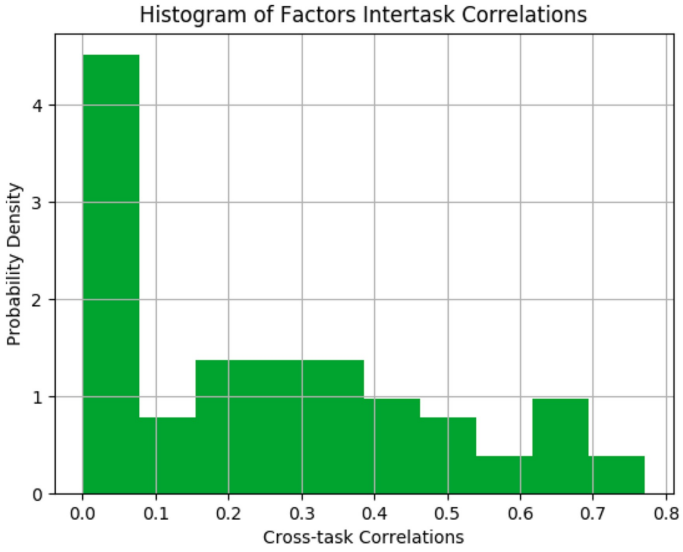
- Only showing results for Protease inhibitors/Factors:

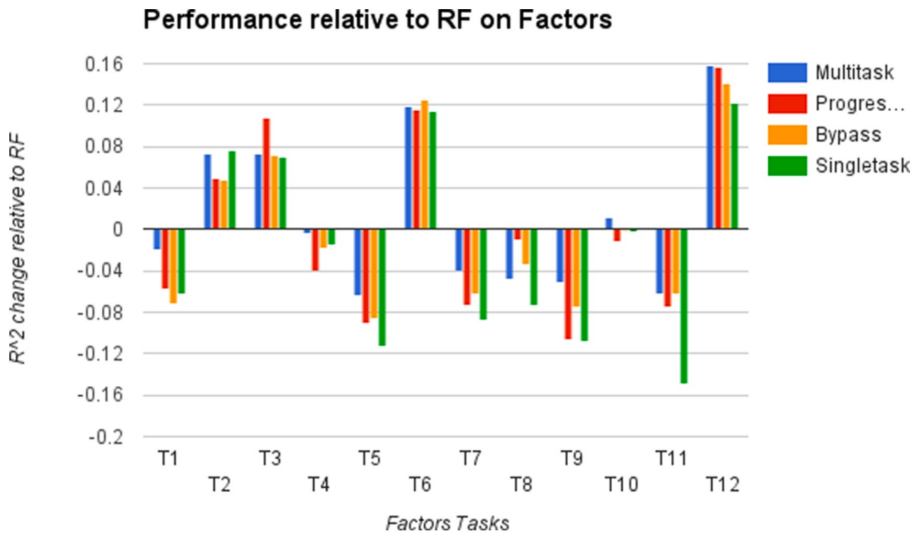
  i) Low task correlation

  ii) Data scarce but dense (1,500 compounds all measured on each assay)

Table 4. Model Performances of the Factors Collection[a]

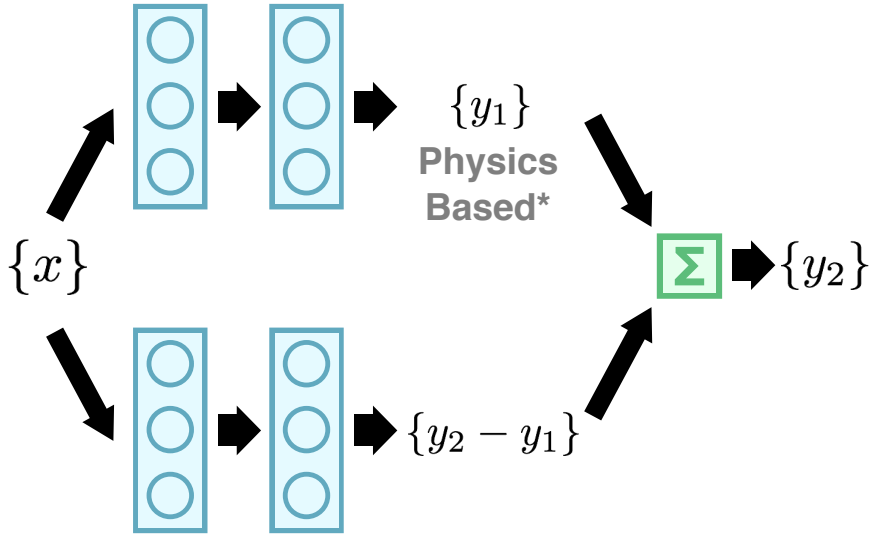
	mean training R^2	mean validation R^2	mean test R^2
multitask	0.941 ± 0.003	0.517 ± 0.014	0.424 ± 0.006
progressive	0.935 ± 0.001	0.499 ± 0.001	0.409 ± 0.005
bypass	0.949 ± 0.001	0.508 ± 0.004	0.410 ± 0.001
singletask	0.954 ± 0.000	0.504 ± 0.005	0.393 ± 0.003
RF	0.949 ± 0.000	0.466 ± 0.005	0.412 ± 0.006





Difference Learning - Concept

Typical Architecture:



Concept:

Learning all of physics

$$\mathbf{X} \xrightarrow{f(\mathbf{X})} \mathbf{Y}_2$$

Just learning the missing physics

$$\mathbf{X} \xrightarrow{g(\mathbf{X})} \mathbf{Y}_2 - \mathbf{Y}_1$$

*top branch might be replaced with physics-based model

When you would use it:

1) You already have one or more decent surrogate models

2) You have a large data disparity. E.g., lots of data for a correlated property but only a small amount of data for a target property.

Hutchinson, M. L.; Antono, E.; Gibbons, B. M.; Paradiso, S.; Ling, J.; Meredig, B. Overcoming Data Scarcity with Transfer Learning. arXiv, 2017.

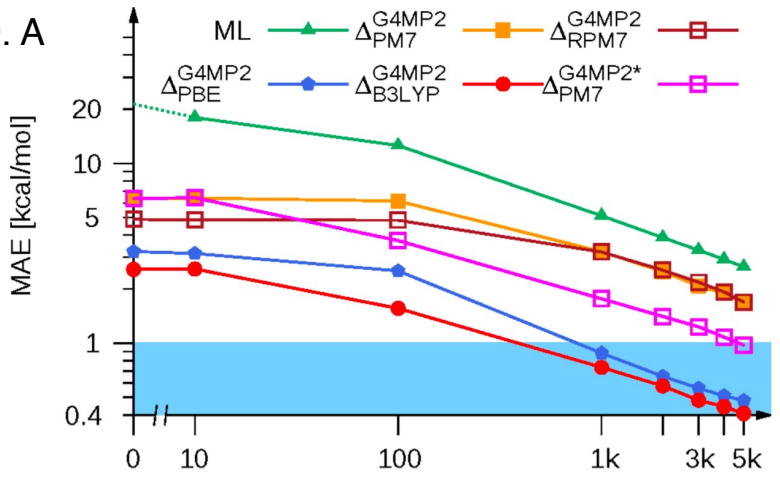
Difference Learning - Example

Lilienfeld Δ-Learning:

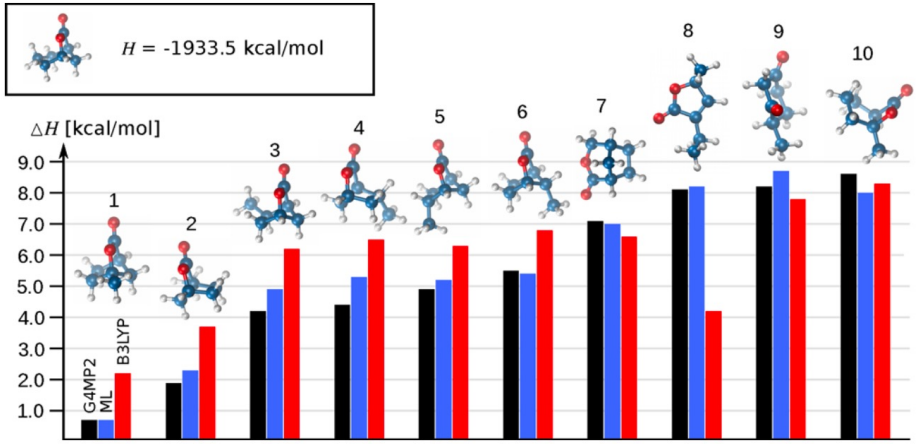
Ramakrishnan, R.; Dral, P. O.; Rupp, M.; von Lilienfeld, O. A *J. Chem. Theory Comput.* **2015**, *11* (5), 2087–2096.

- Order of magnitude improvements over trying to directly learn the high-level relationship (green curve vs others)
- Depends on quality of surrogate model
- Translates across prediction tasks

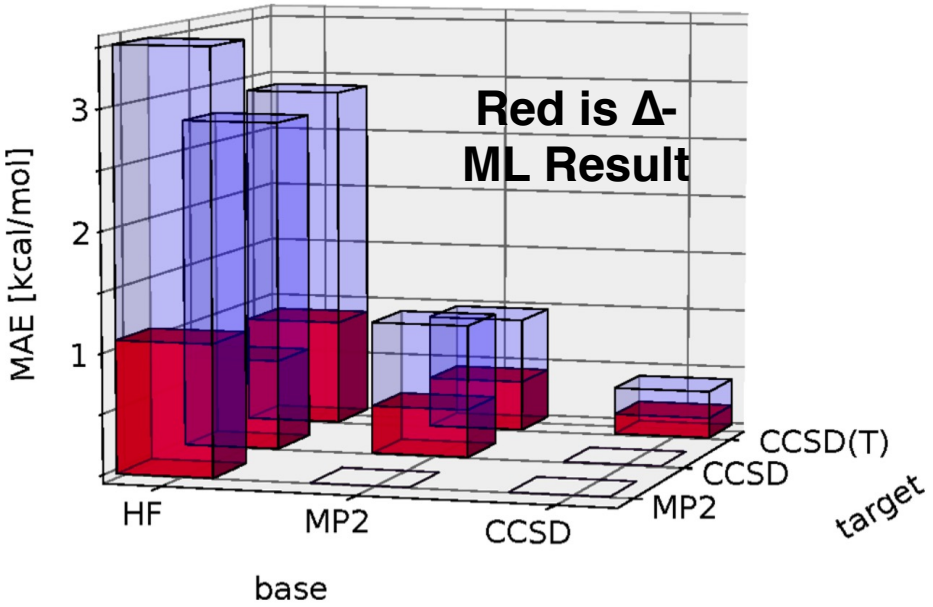
Atomization Energy Task



Reaction Enthalpy Prediction Task

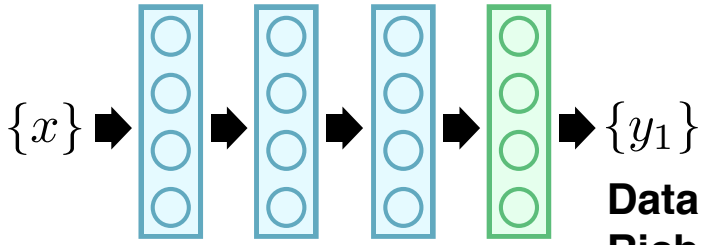


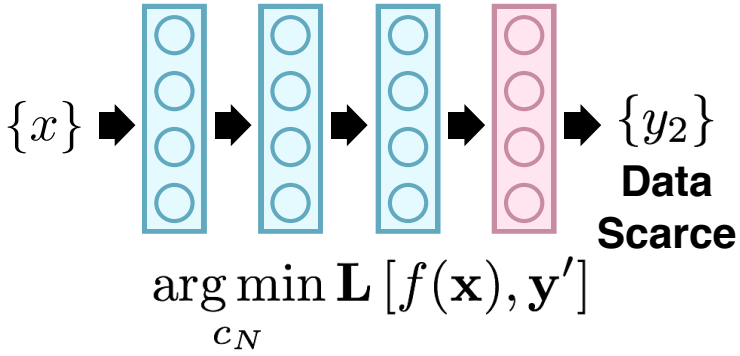
Black: Reference; Blue: Δ-1k; Red: B3LYP



Retraining Learning - Concept

Typical Architecture:





Concept: **Low level representation is transferable between tasks** ➡ **Just retrain the high-level predictor, don't relearn everything**

When you would use it:

1) You already have great model for a given **x** → **y** task

2) You now want a model for **x→y'** where **x** are common features between tasks.

3) Different from multitask in that there is no mutual improvement or regularization; the transfer is all downstream.

Retraining Learning - Example

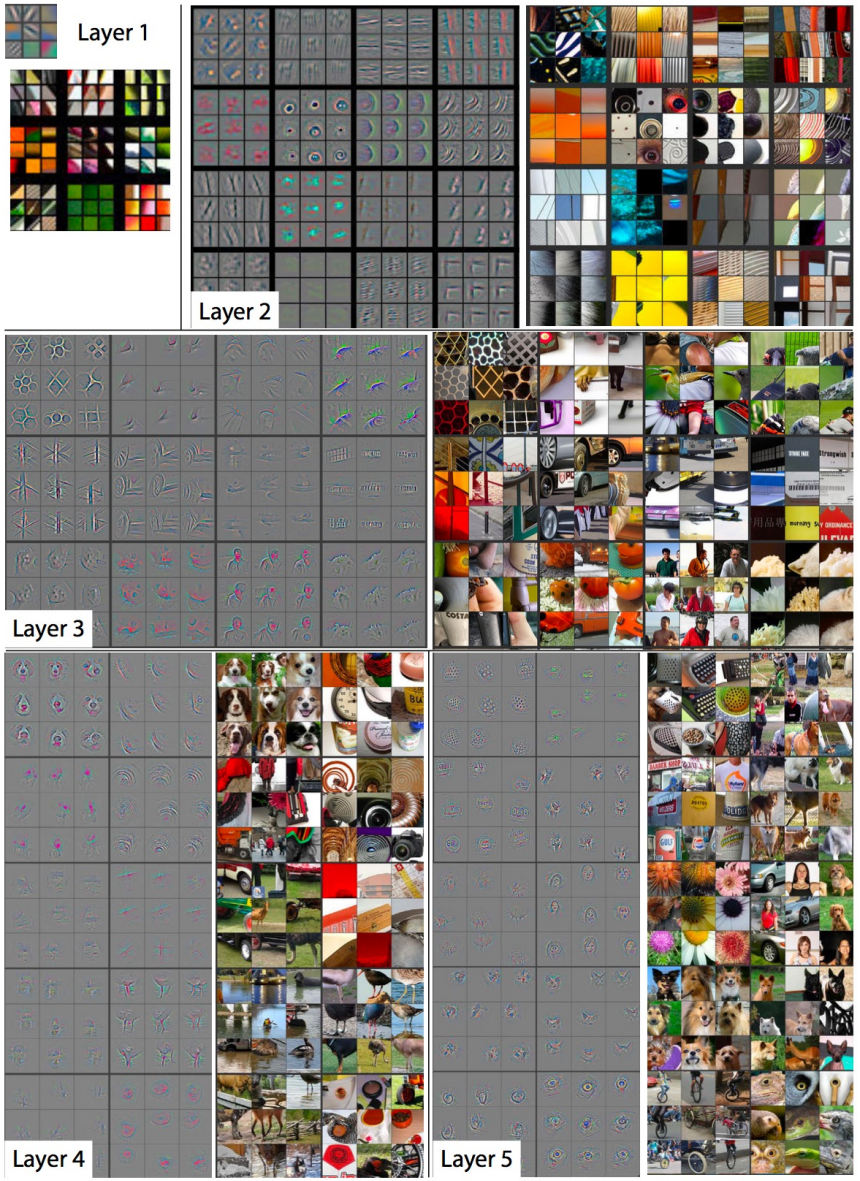
Image Classification (Imagenet Task):

- This is a stock example for retraining: Lots of data available for images, potentially few images available for specific domains.

- Base layers of deep convolutional classifiers learn generic features like edges and gradients that are transferable to other image classification tasks.

- Later layers learn more specific patterns for the training task/data.

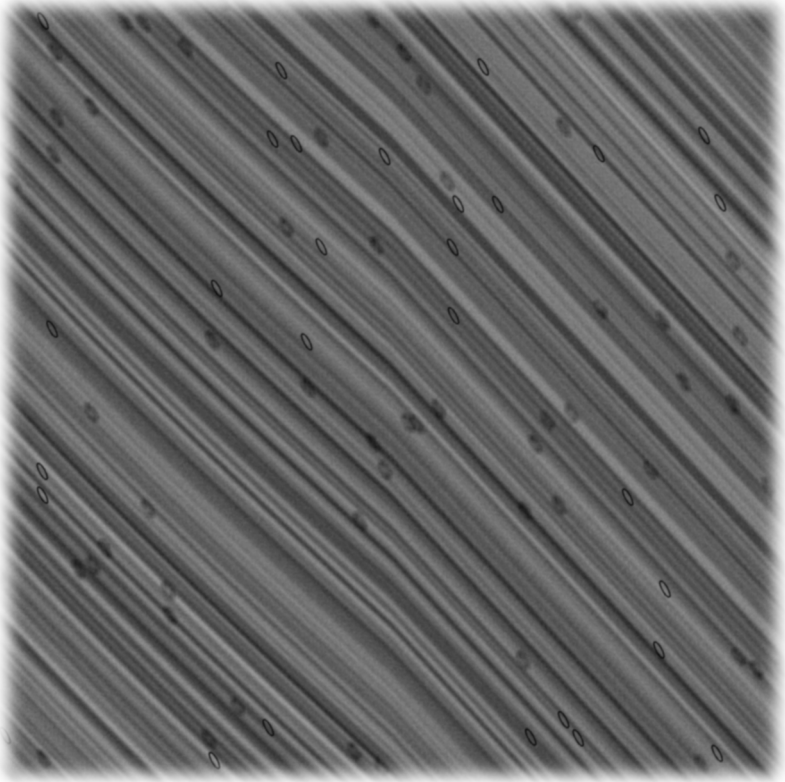
The idea is to utilize this rich latent representation in a domain specific image classification task.



Zeiler, M. D.; Fergus, R. Visualizing and Understanding Convolutional Networks. *arXiv:1311.2901* **2013**.

Retraining Learning - Example

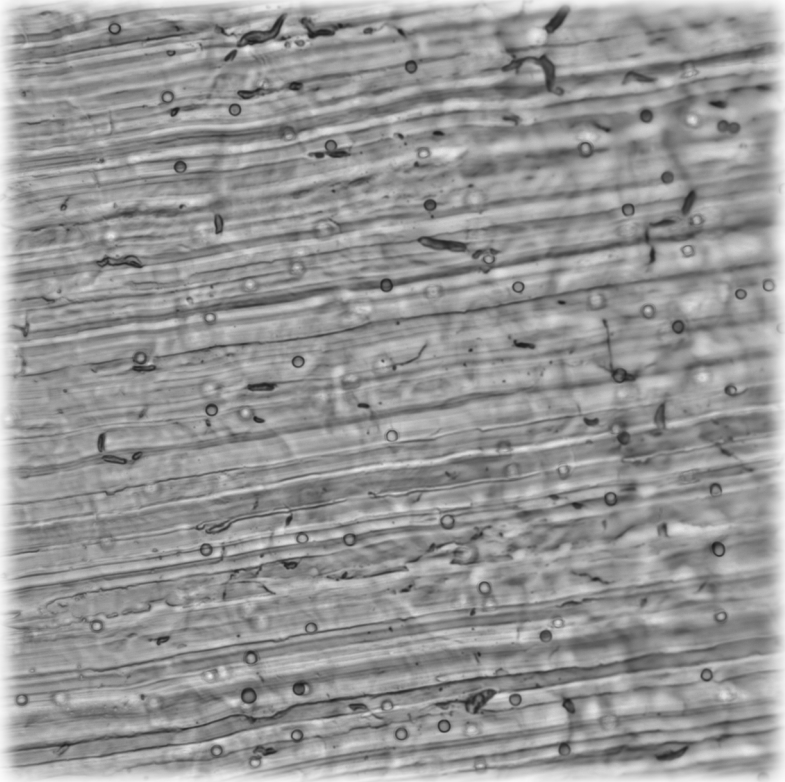
Data Scarce Problem (particle identification in non-Newtonian flow):



Synthetic Data

Collection Rate: ~Infinite

Ground Truth Labels: Obtained at generation



Experimental Data

Collection Rate: ~100 per month

Ground Truth Labels: Undefined

Retraining Learning - Example

Utilize a pretrained object segmentation classifier for our task (Mask R-CNN)

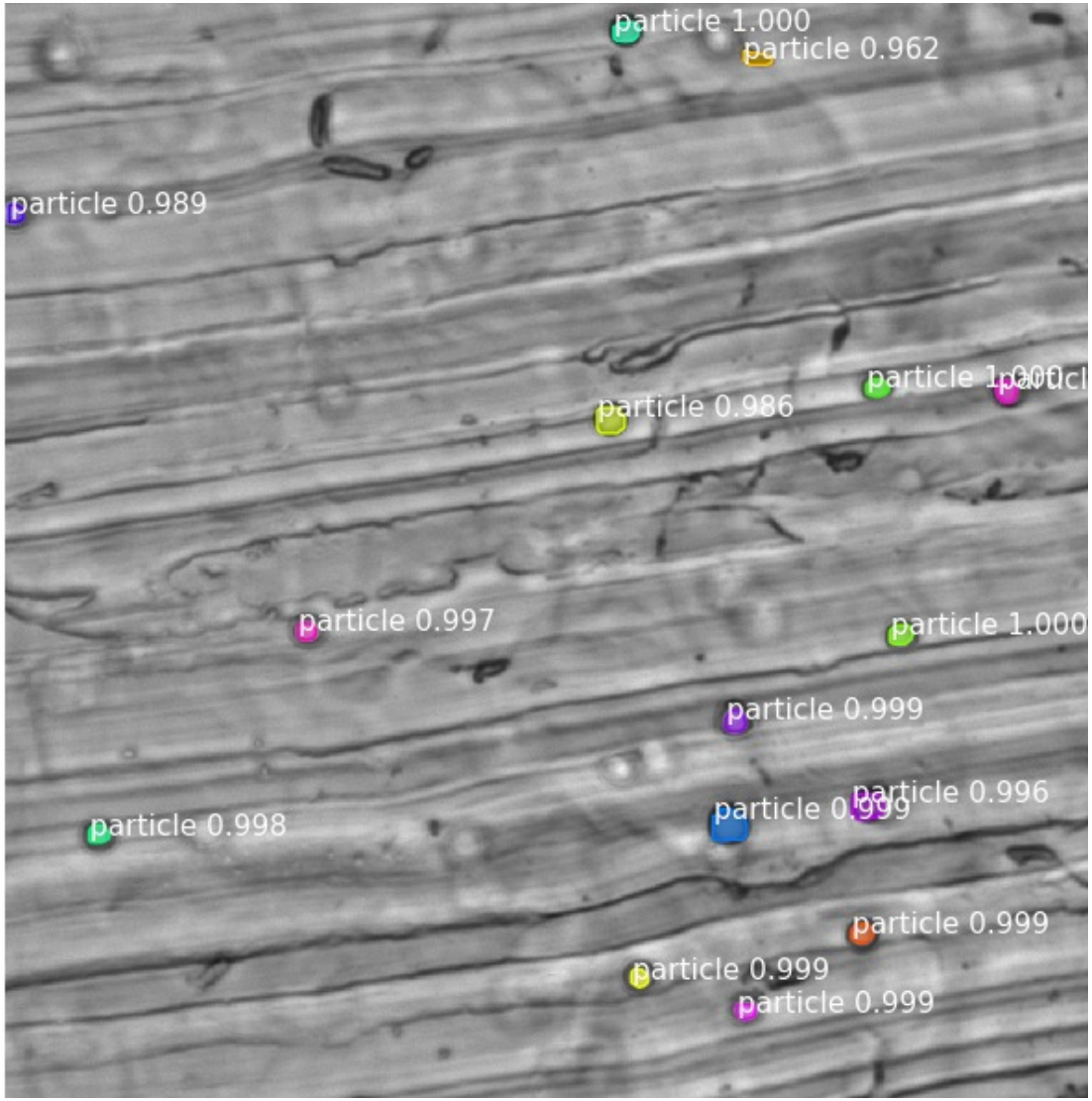
Kaiming He; Georgia Gkioxari; Piotr Dollar; Ross Girshick. Mask R-CNN. *ICCV* **2017**.

We've retrained the last layer on our synthetic dataset then tested it on the experimental data (double transfer learning!)

Retraining Learning - Example

Model returns intuitive masks for the particles in the unseen experimental test data.

Building this model from scratch would have taken many months, but it only took Nick Iovanac a couple days utilizing the pretrained model.



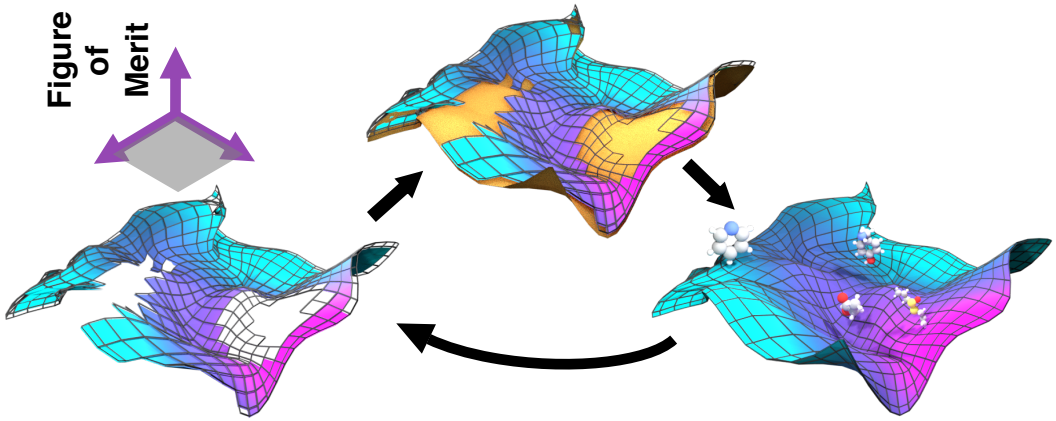
*Out of focus particles are missed by design

Outline

Transfer Learning:

Paradigms with examples:

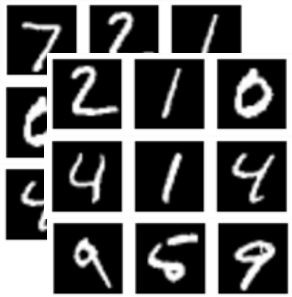
Contemporary Topics:



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

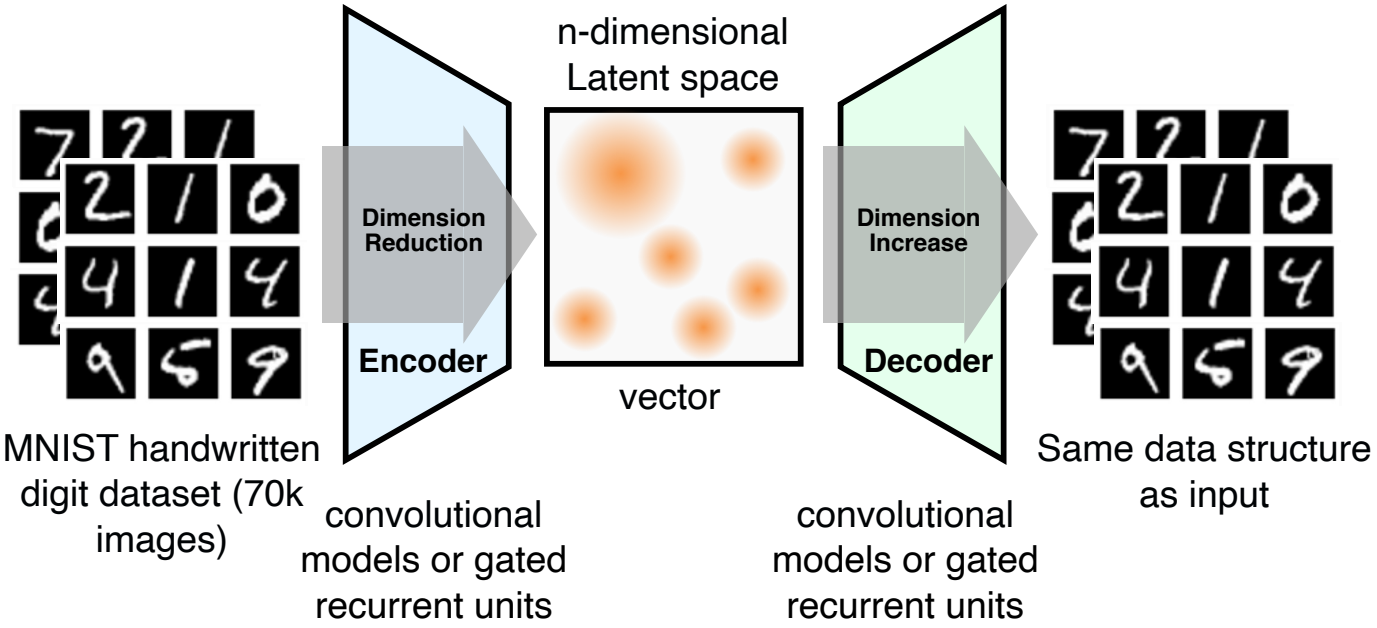
2) Encode: $f(\phi_{in}) = \mathbf{r}$

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

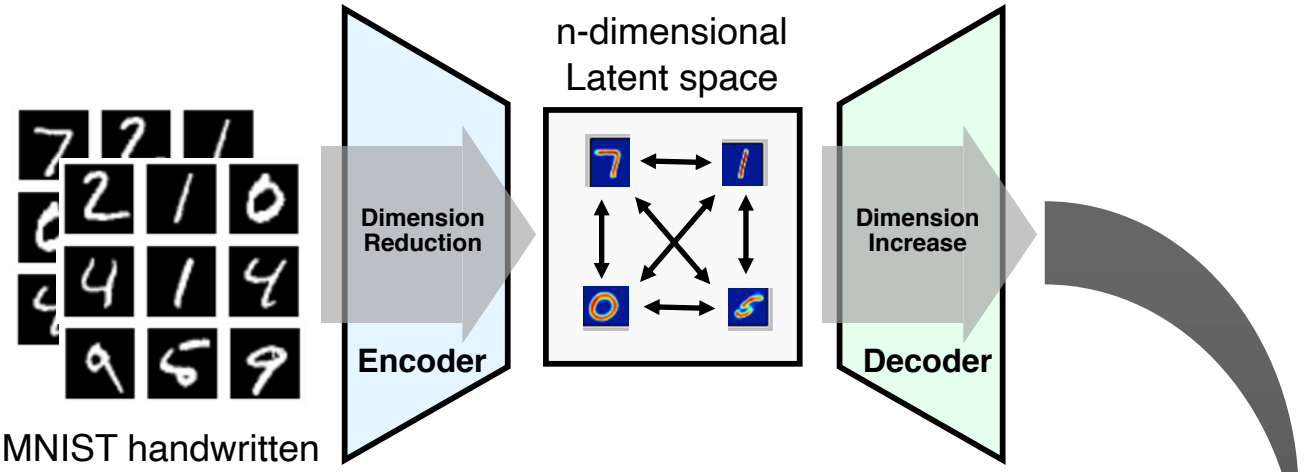
2) Encode: $f(\phi_{in}) = \mathbf{r}$ $\quad f(\phi_{in}) = (\boldsymbol{\mu}, \boldsymbol{\sigma})$

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

4) Train on reconstruction loss

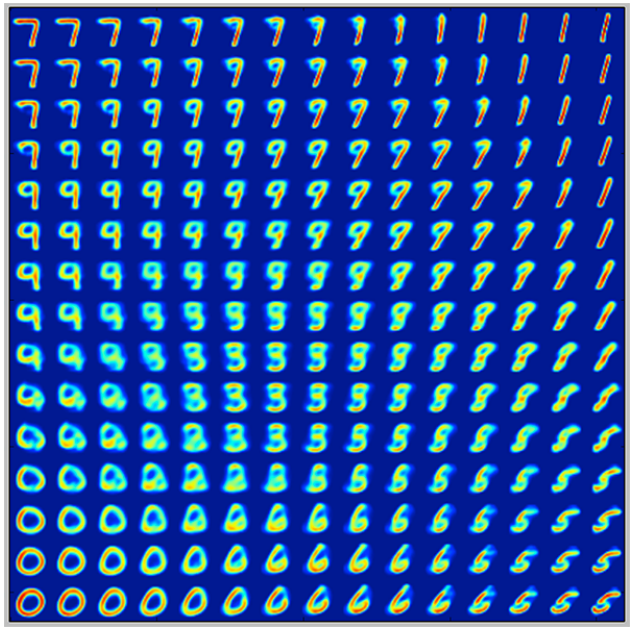
Generative Models Are Neat

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



MNIST handwritten digit dataset (70k images)

After training, the latent space can be explored and decoded to "generate" new data



Corners are the only "real" images

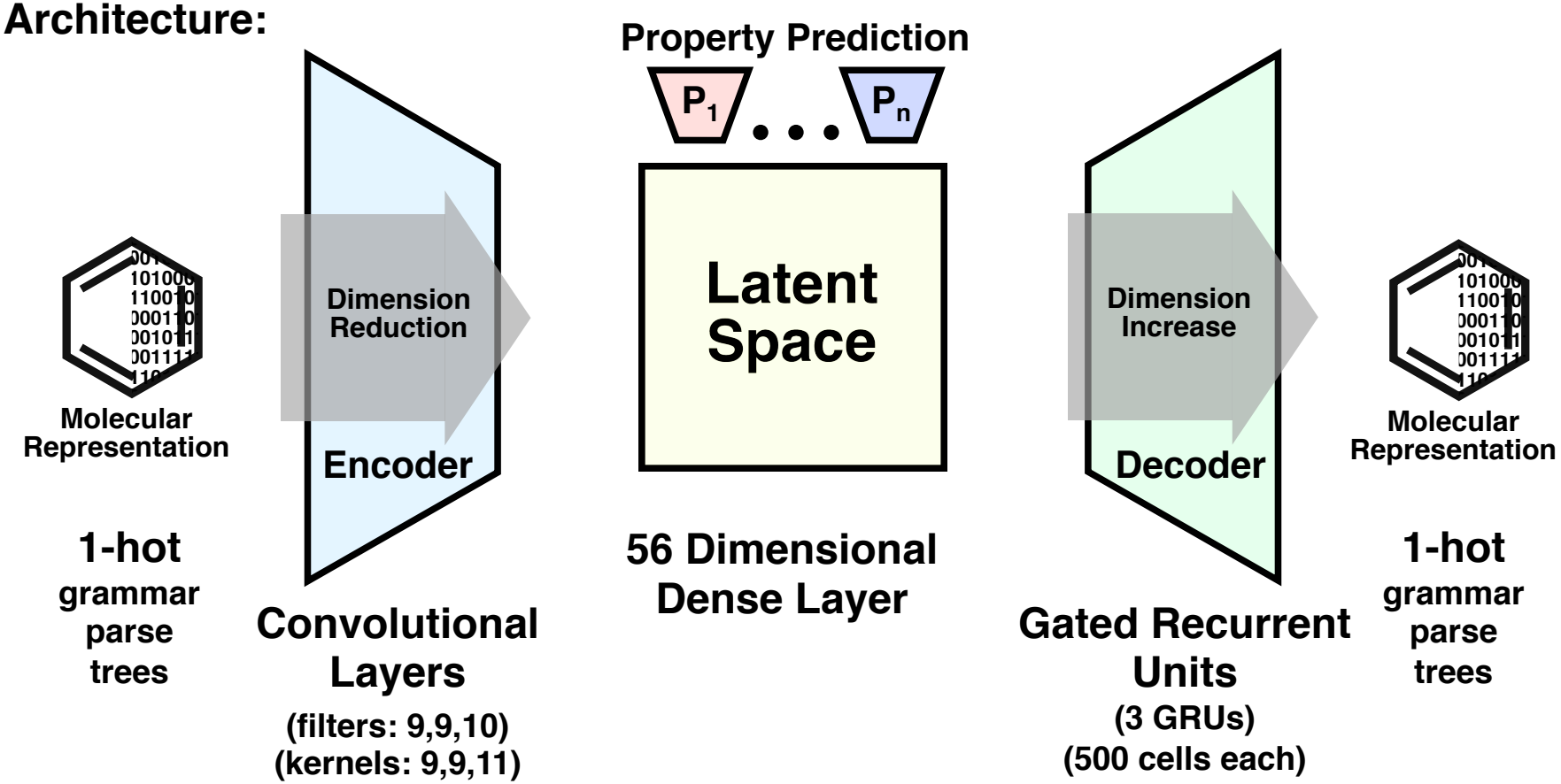
Transfer Learning for Generative Chemical Models

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

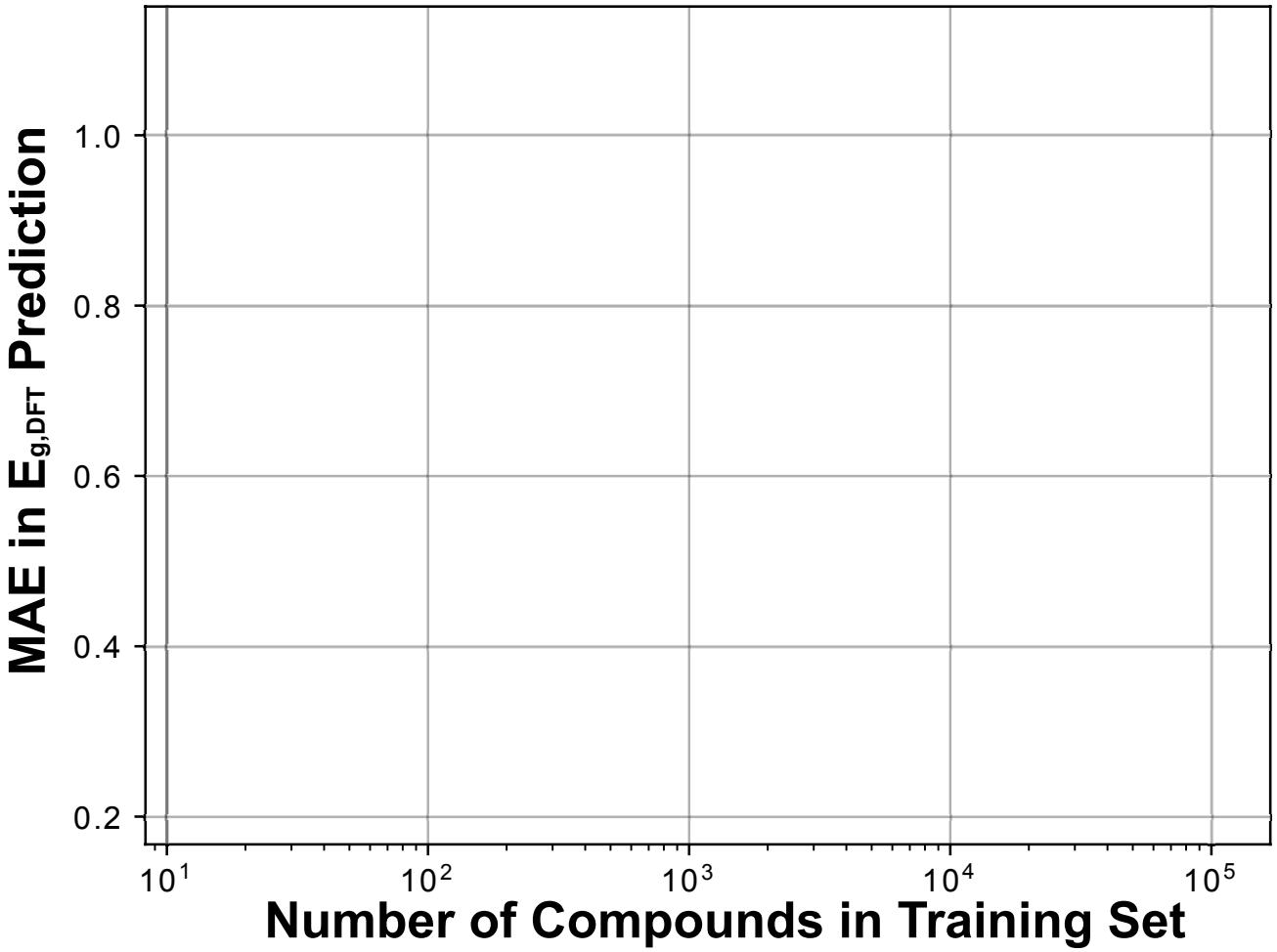
This is effectively a multitask transfer learning activity, **but for a model that is capable of generating compounds.**

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**

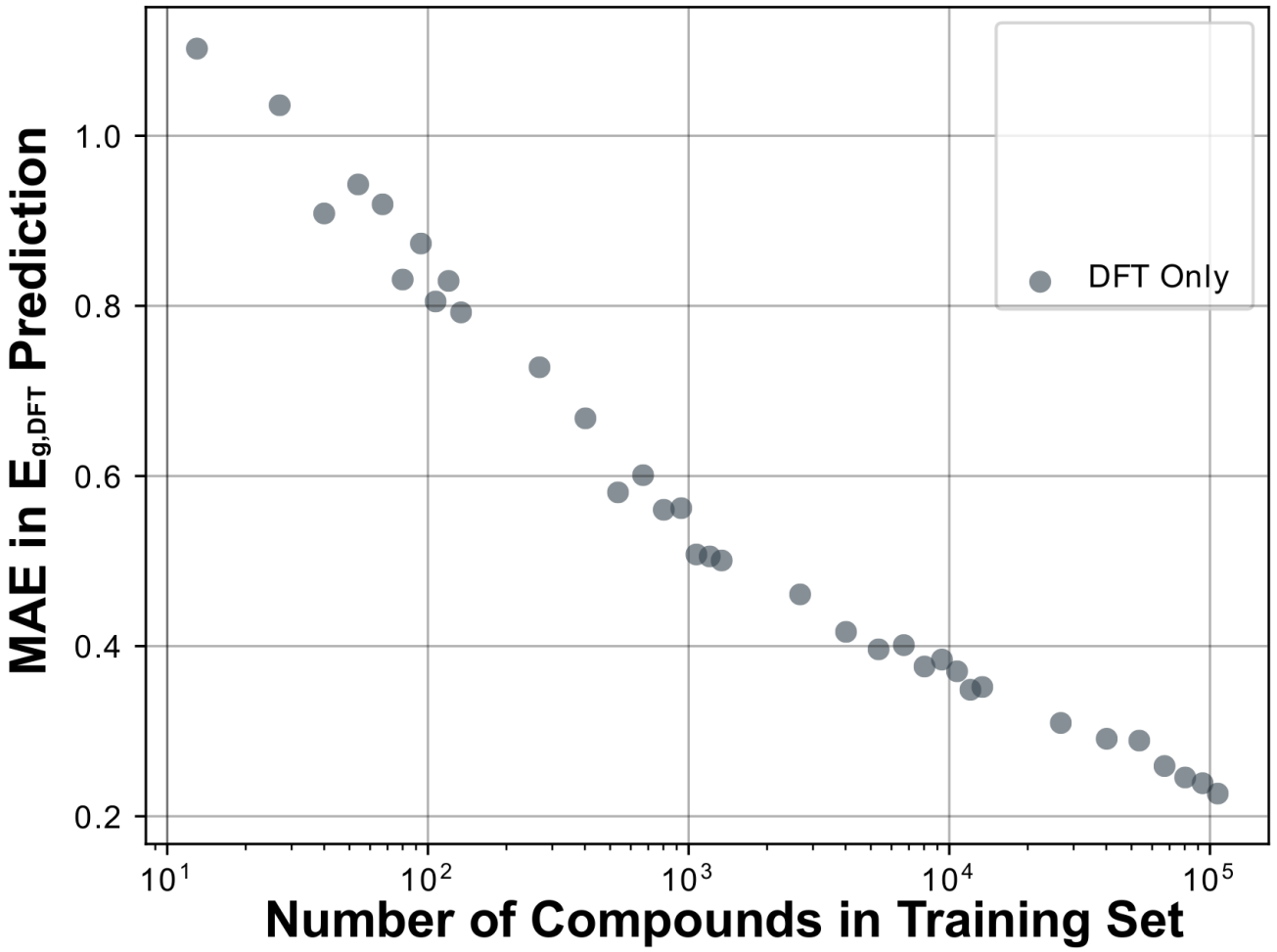
Transfer Learning on the Molecular Bandgap



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

Data Scarce: DFT bandgap prediction (hours per compound)

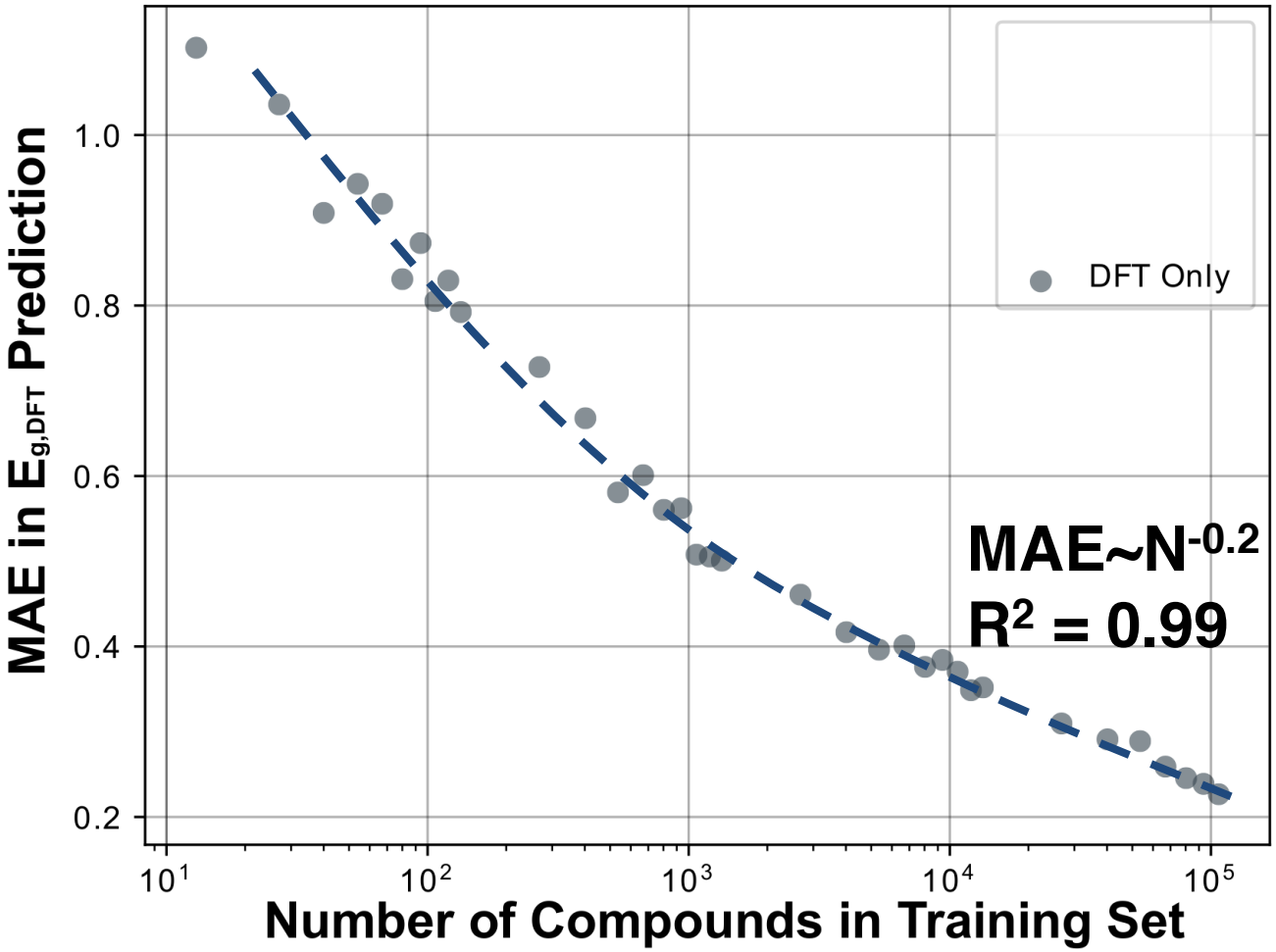
Transfer Learning on the Molecular Bandgap



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

Data Scarce: DFT bandgap prediction (hours per compound)

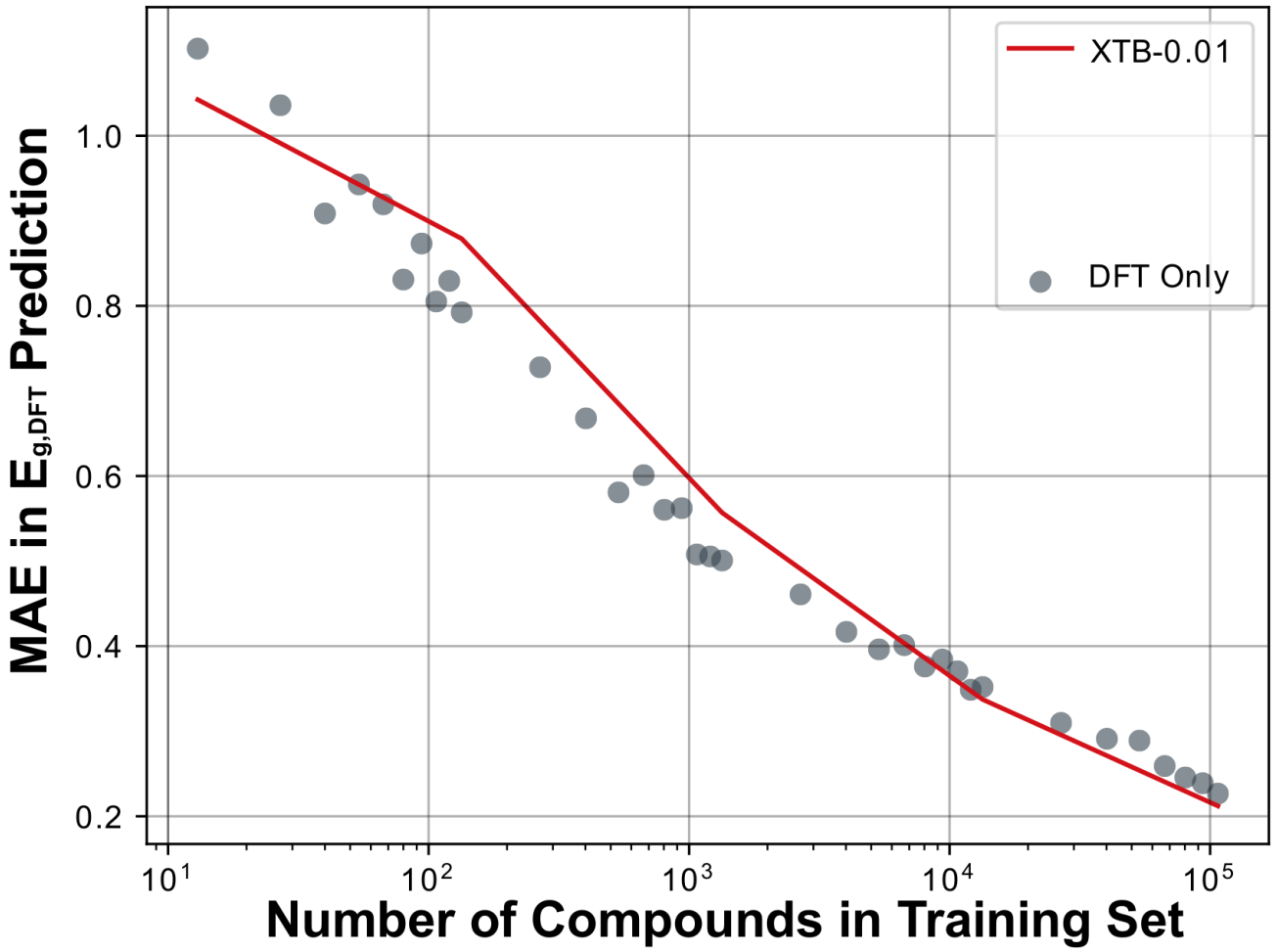
Transfer Learning on the Molecular Bandgap



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

Data Scarce: DFT bandgap prediction (hours per compound)

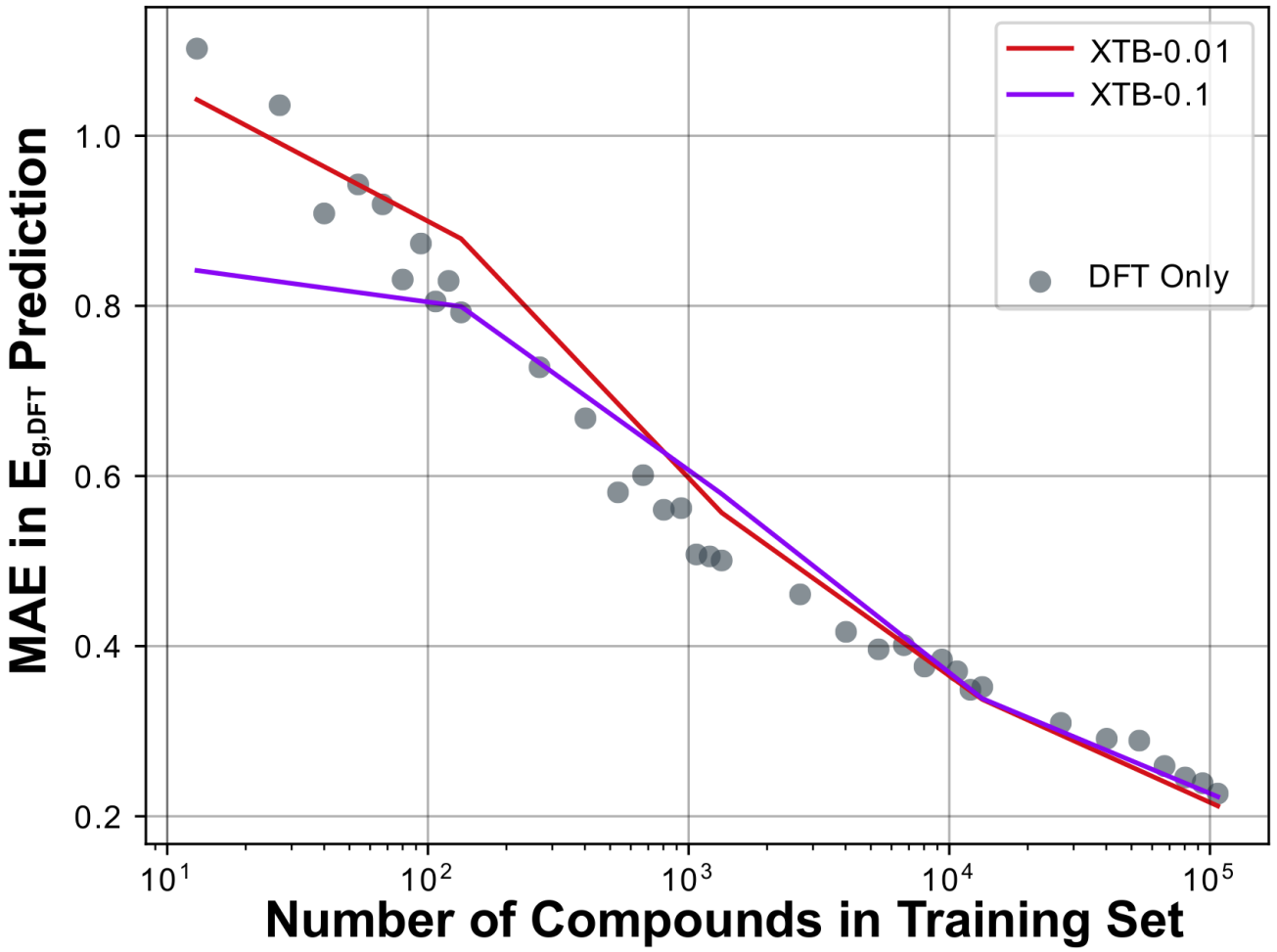
Transfer Learning on the Molecular Bandgap



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

Data Scarce: DFT bandgap prediction (hours per compound)

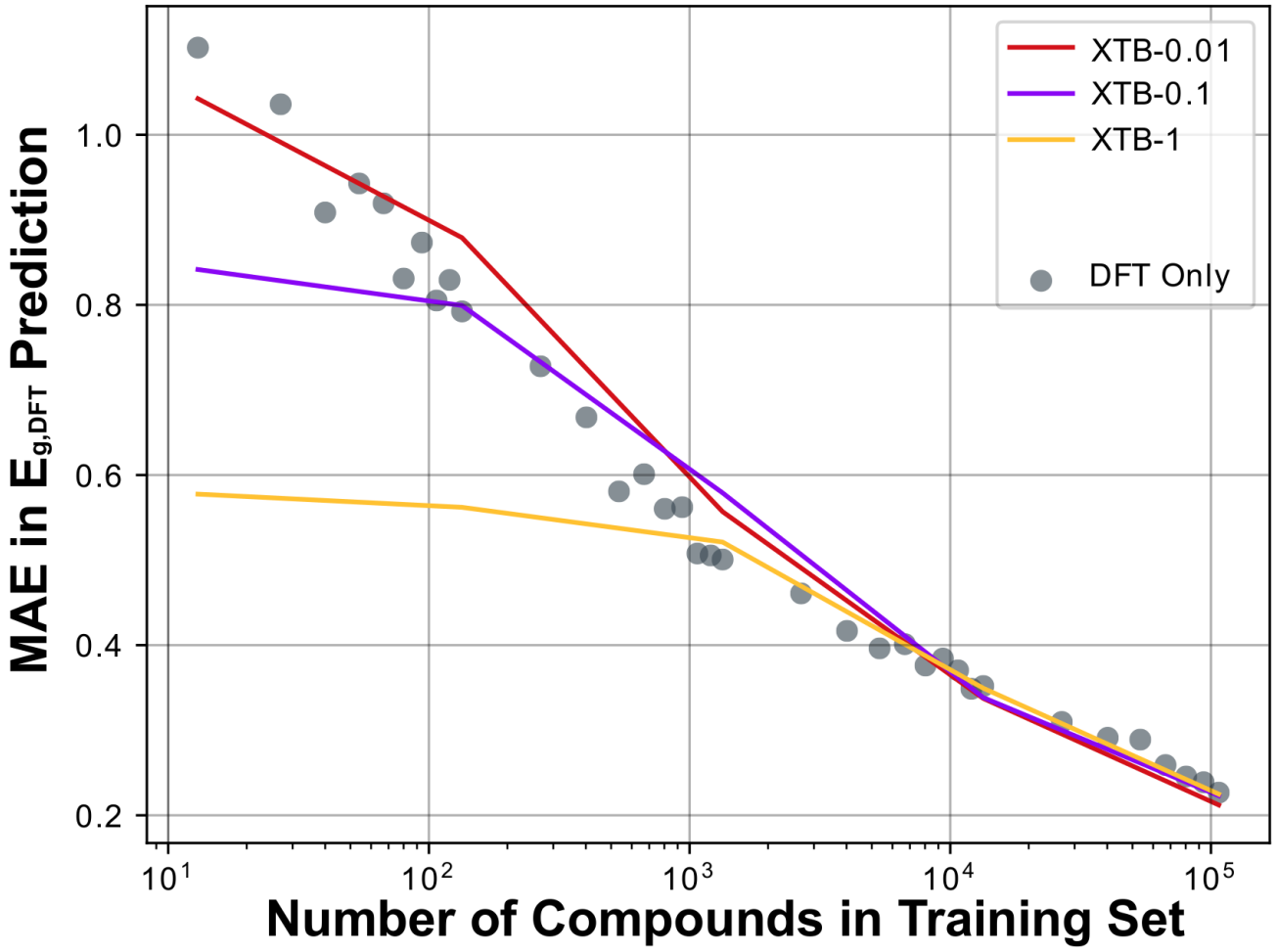
Transfer Learning on the Molecular Bandgap



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

Data Scarce: DFT bandgap prediction (hours per compound)

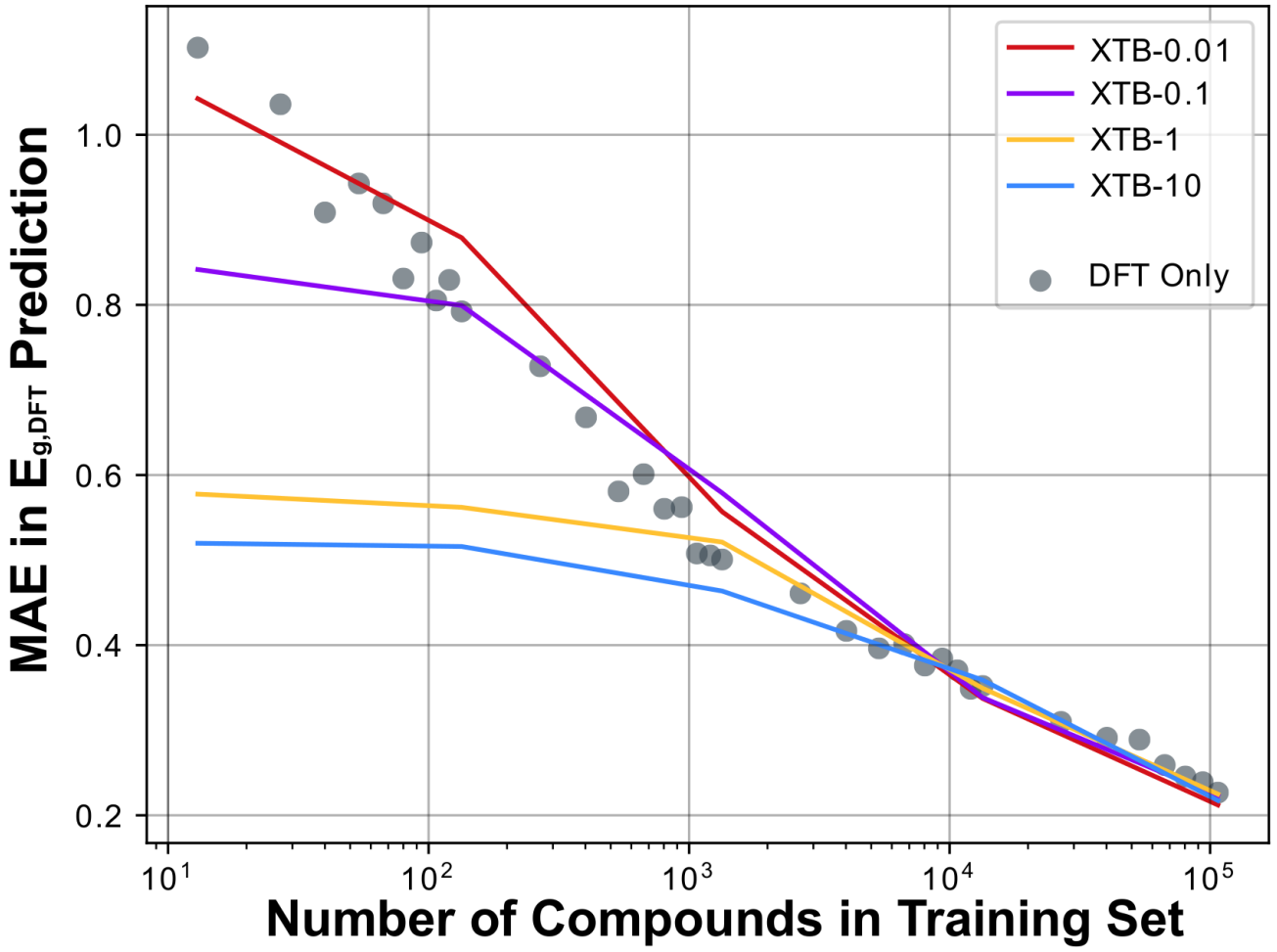
Transfer Learning on the Molecular Bandgap



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

Data Scarce: DFT bandgap prediction (hours per compound)

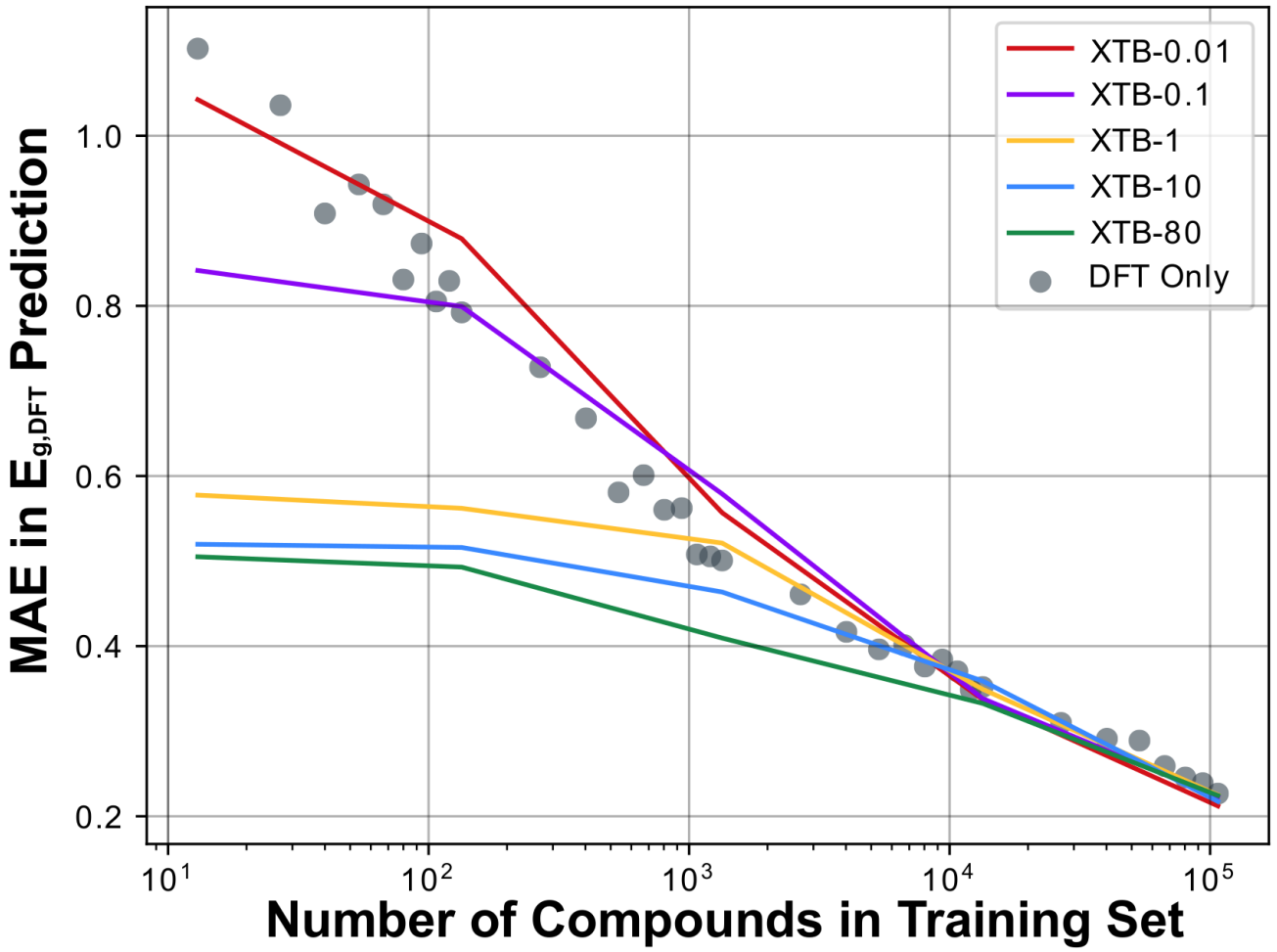
Transfer Learning on the Molecular Bandgap



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

Data Scarce: DFT bandgap prediction (hours per compound)

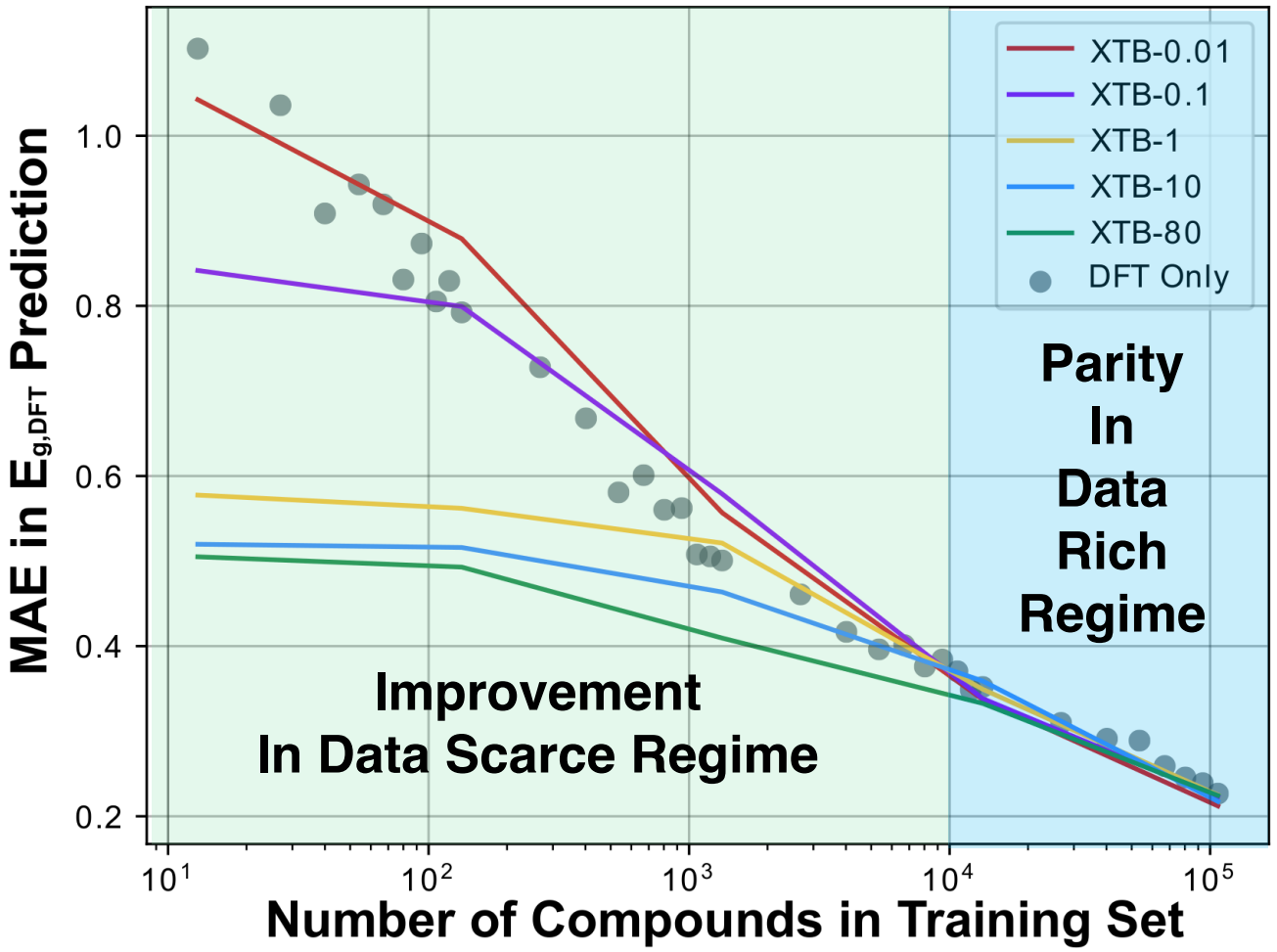
Transfer Learning on the Molecular Bandgap



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

Data Scarce: DFT bandgap prediction (hours per compound)

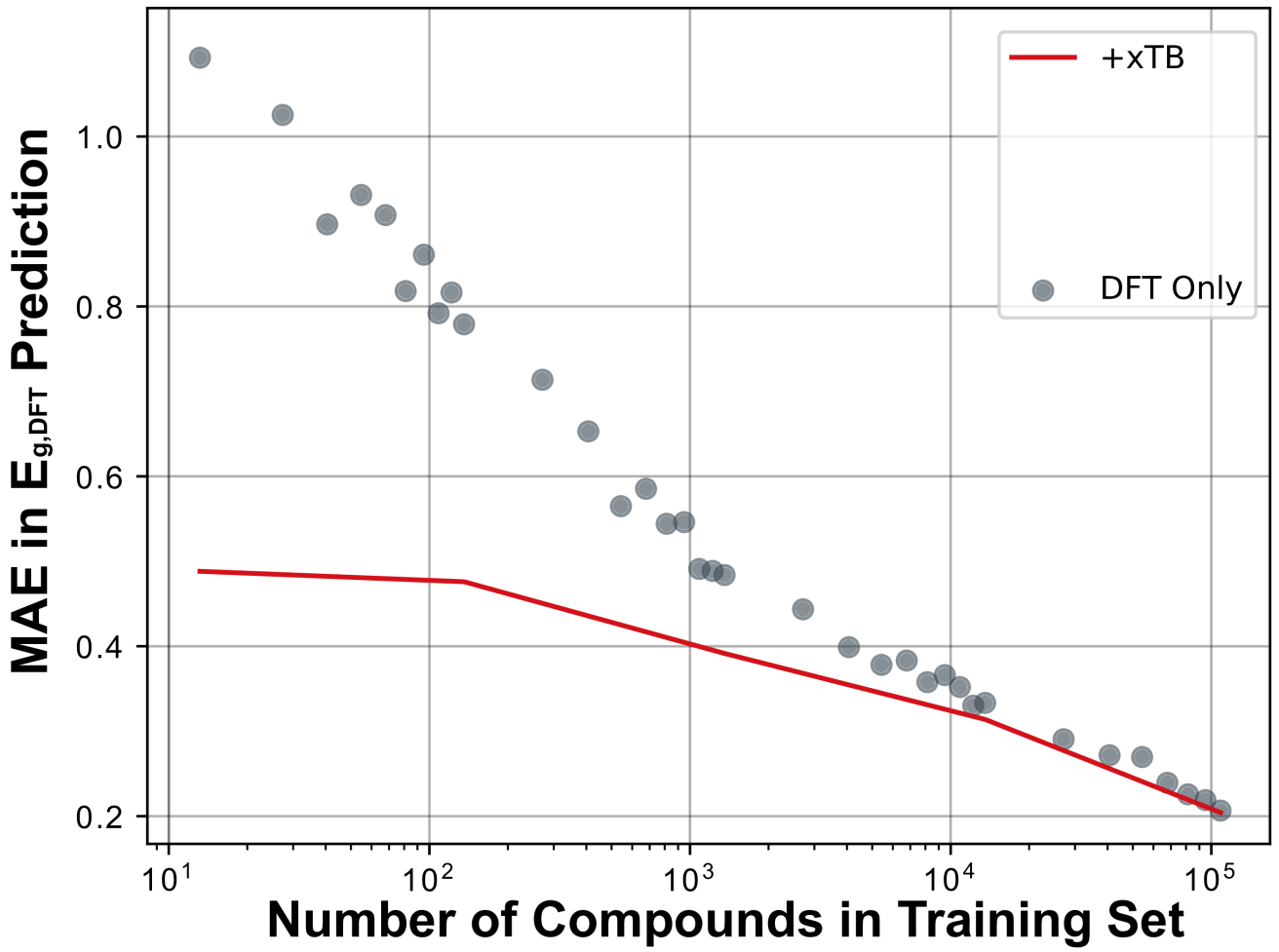
Transfer Learning on the Molecular Bandgap



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

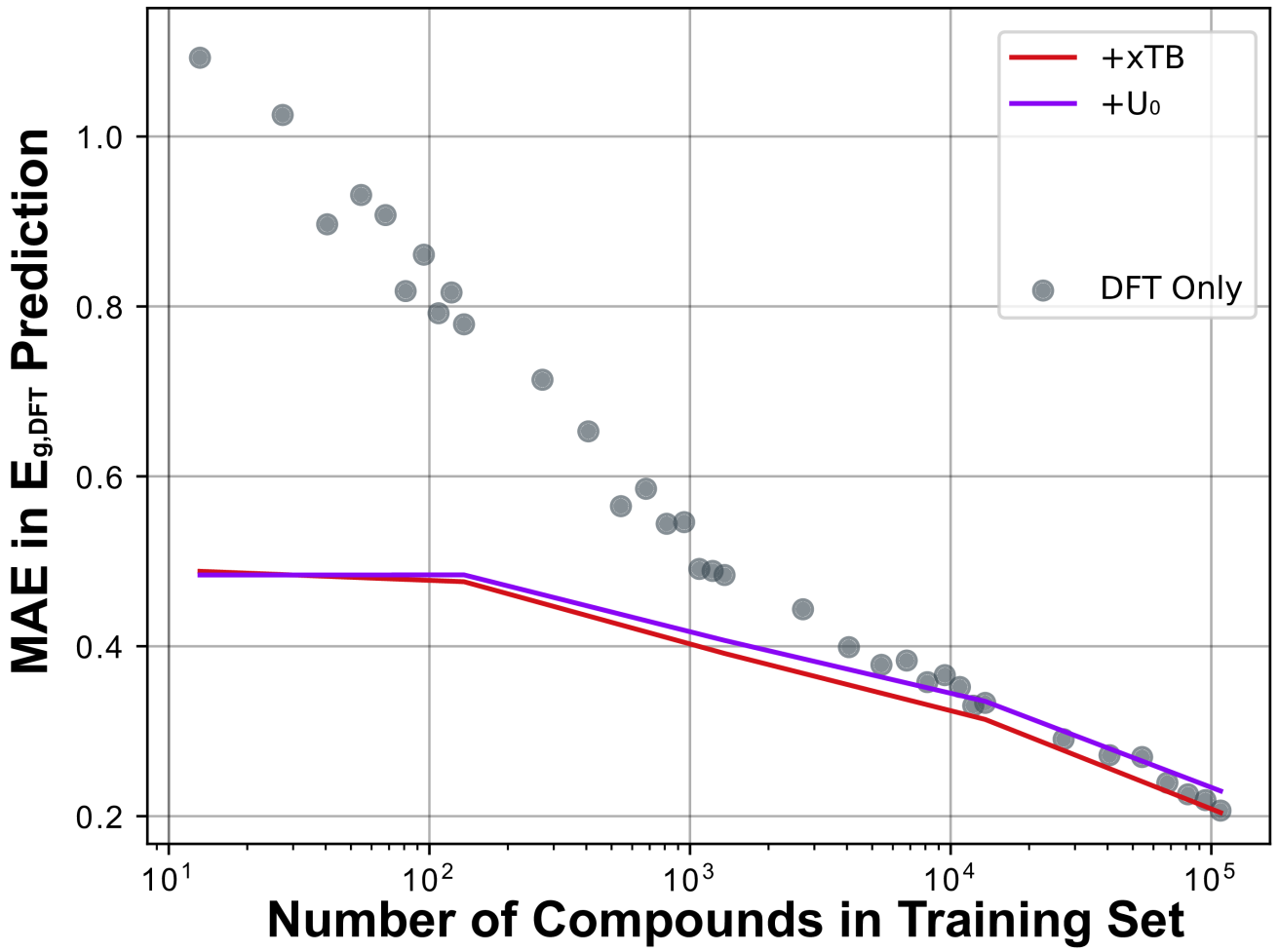
Data Scarce: DFT bandgap prediction (hours per compound)

How Much More Information Can We Add?



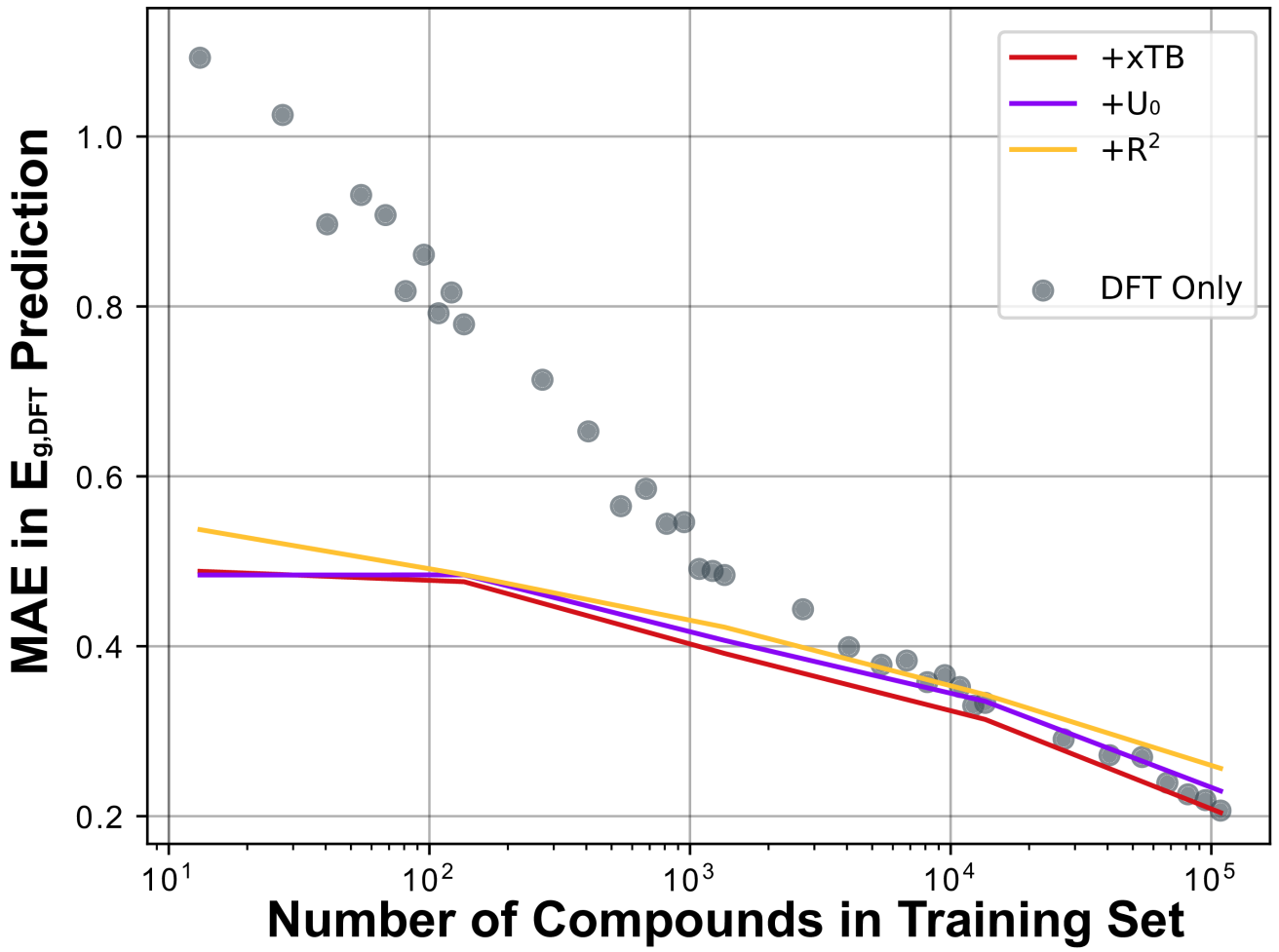
Enrichment with respect to other QM9 molecular properties (full 0.8 training fraction enrichment)

How Much More Information Can We Add?



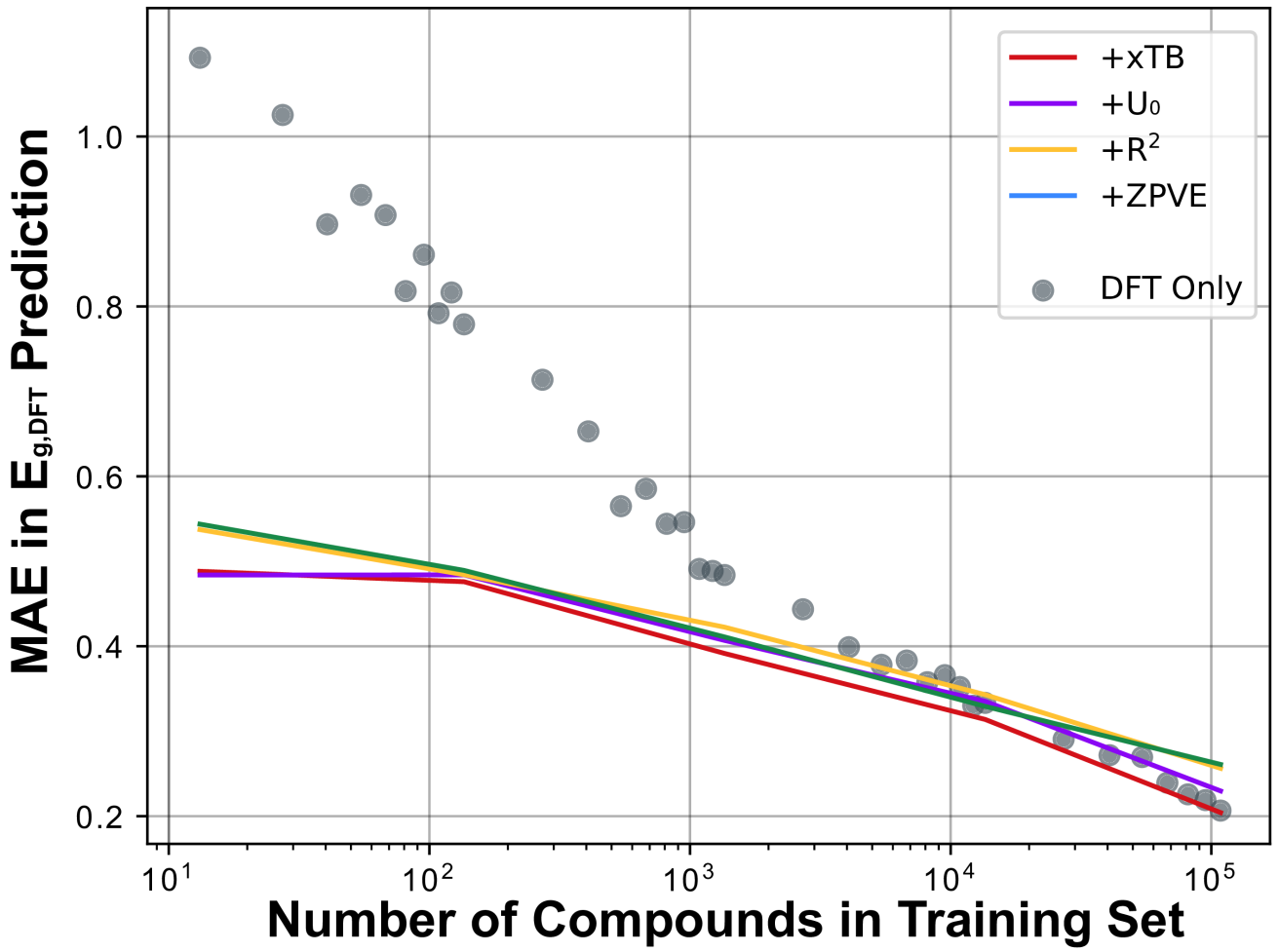
Enrichment with respect to other QM9 molecular properties (full 0.8 training fraction enrichment)

How Much More Information Can We Add?



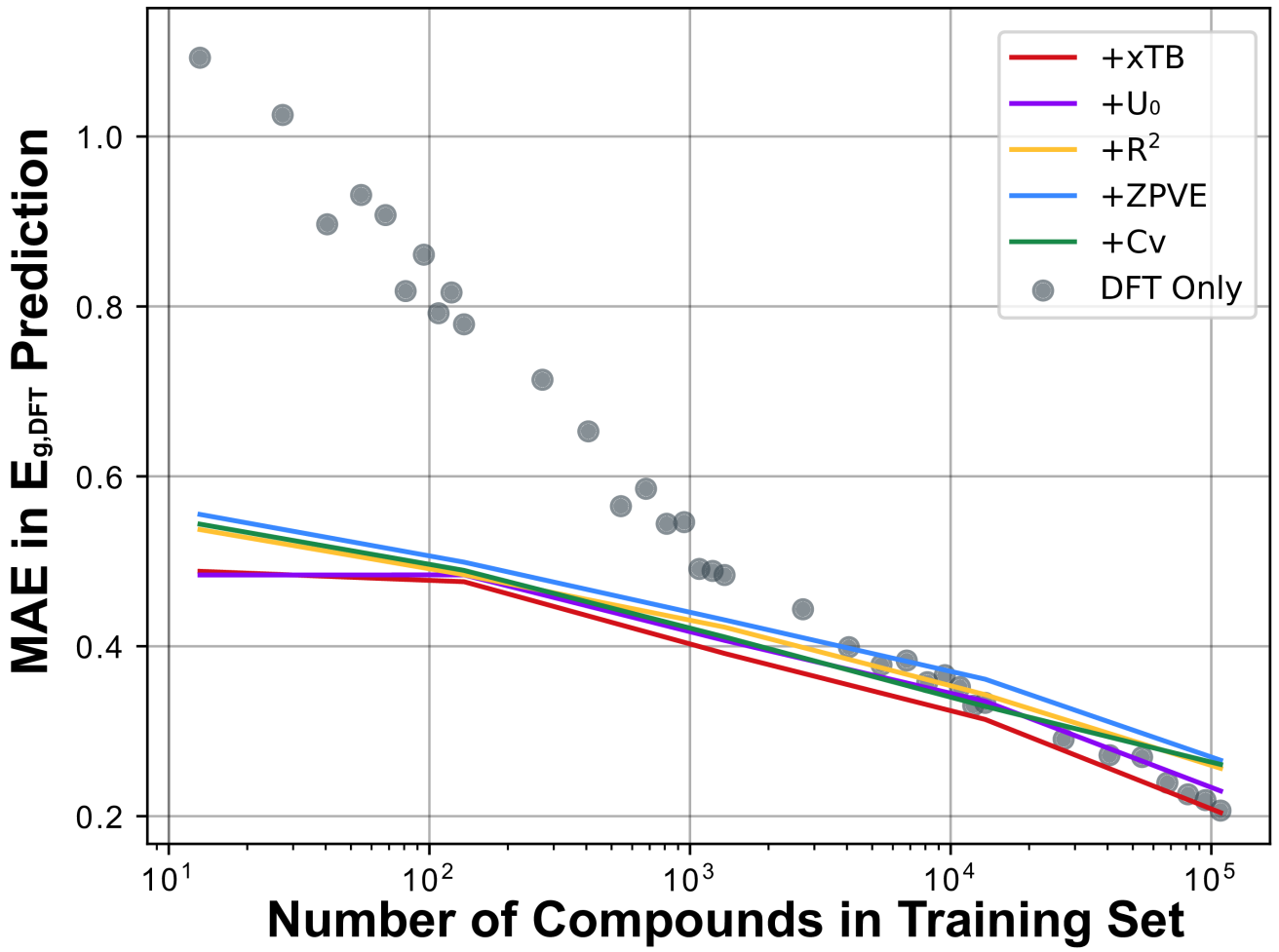
Enrichment with respect to other QM9 molecular properties (full 0.8 training fraction enrichment)

How Much More Information Can We Add?



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

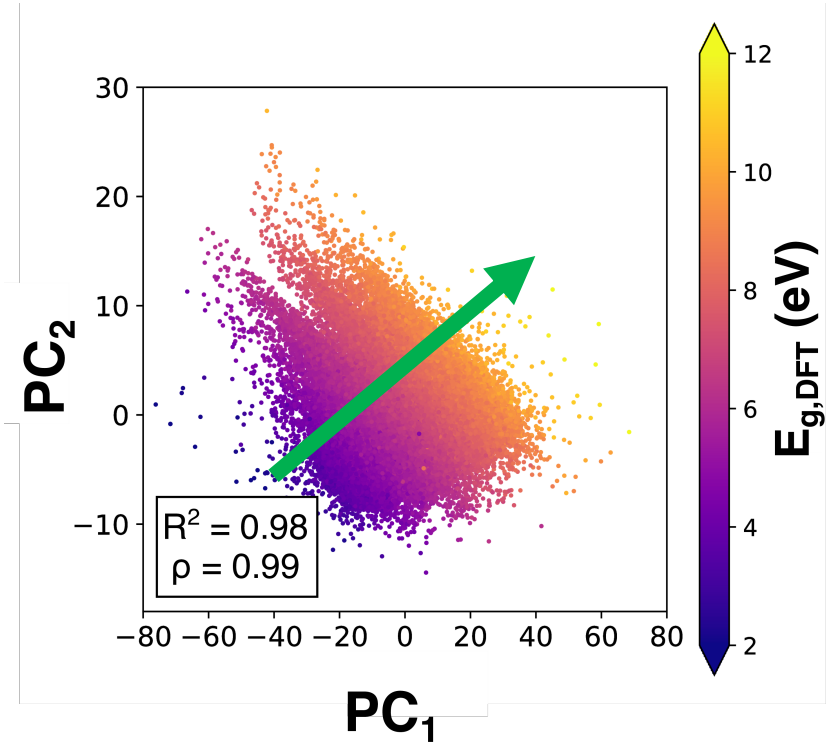
How Much More Information Can We Add?

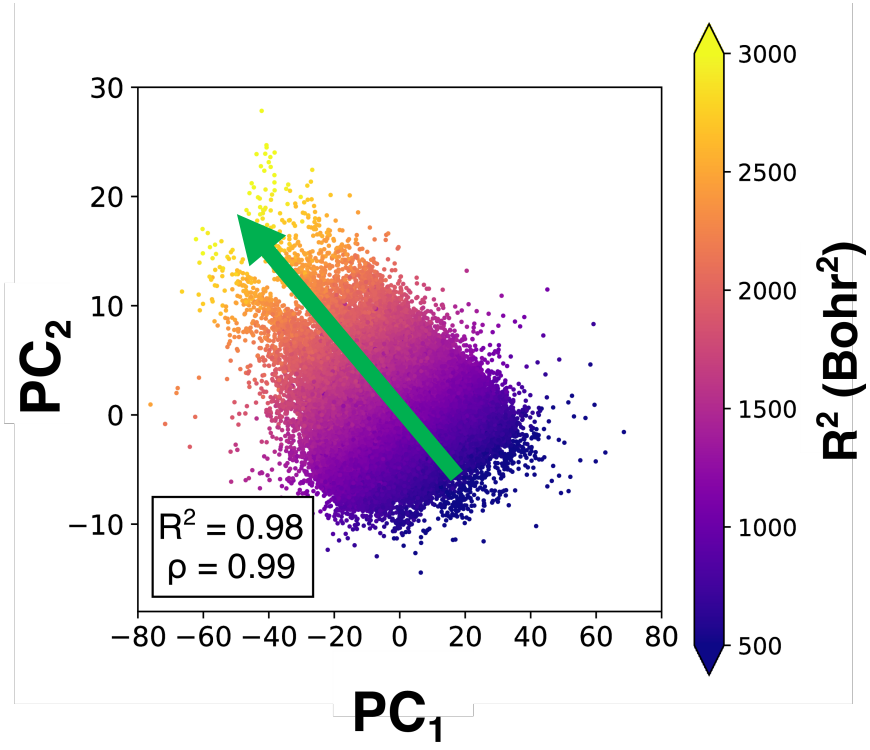


These additional properties are not correlated with bandgap, but they do add physical information and interpretability to the latent space.

Training on Multiple Properties Produces a More Physically Rich Latent Space

Latent Space Encodings for an E_g/R^2 Model

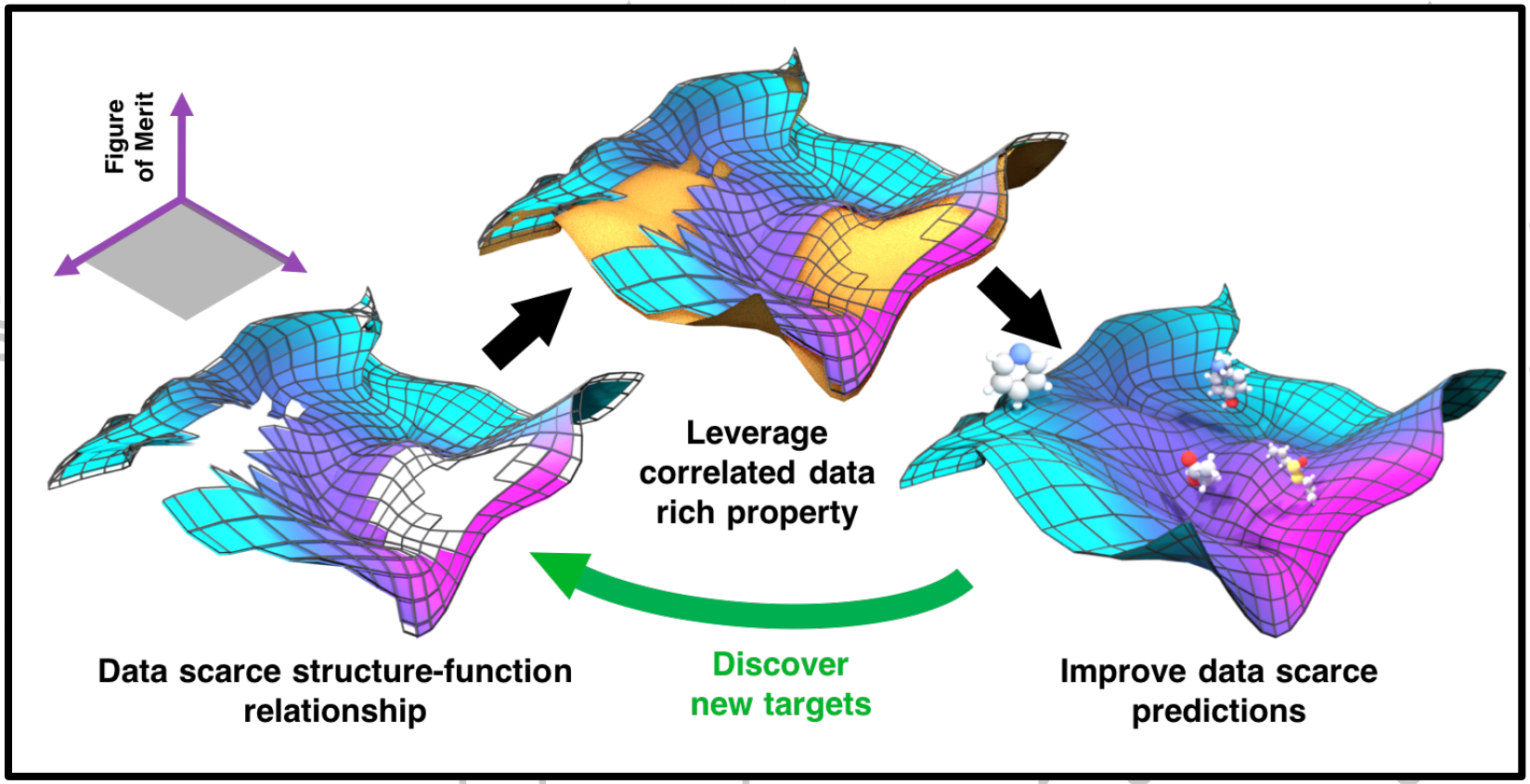




Non-correlated properties spontaneously organize along orthogonal latent space axes

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?

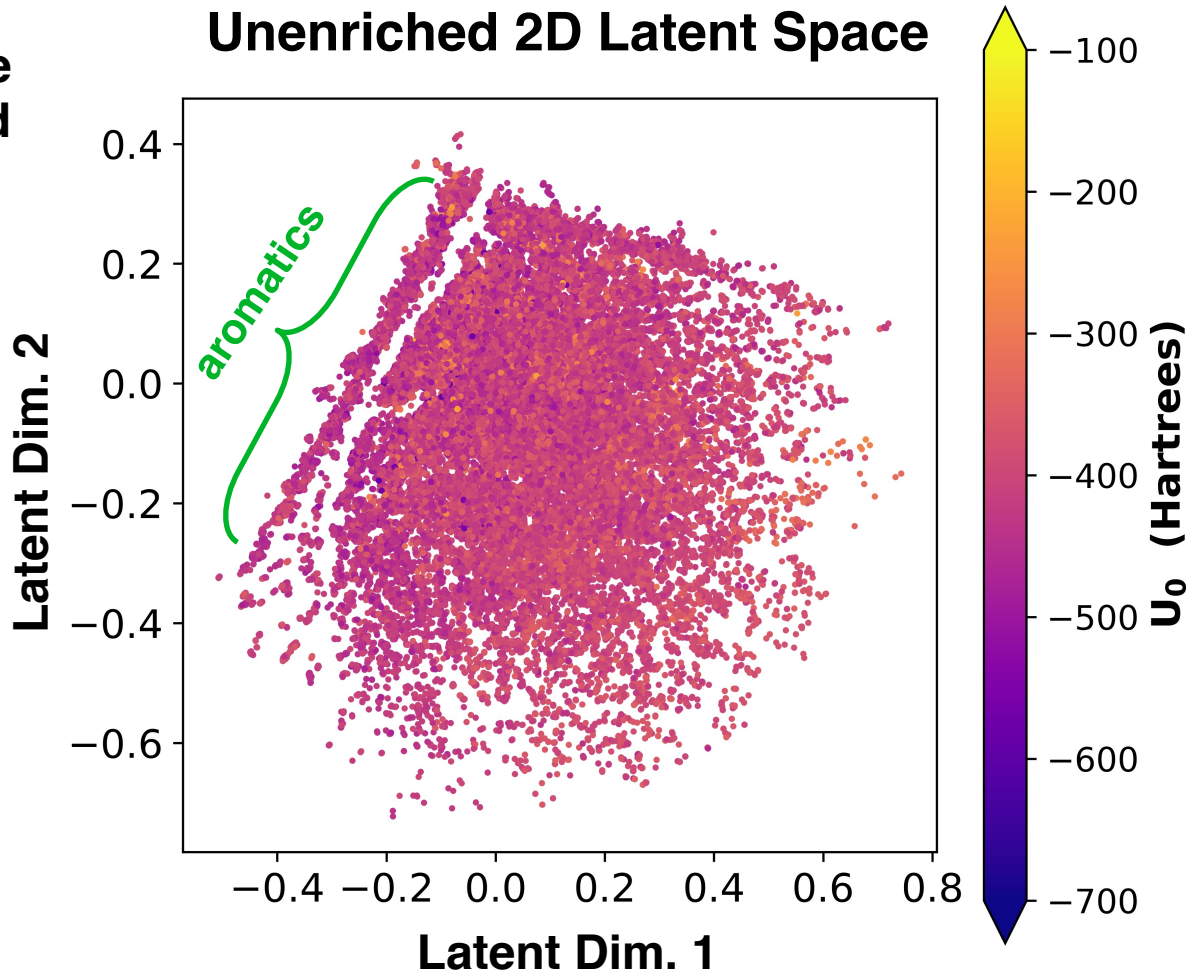
After training only ~4% of the latent space decodes to valid mappings

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



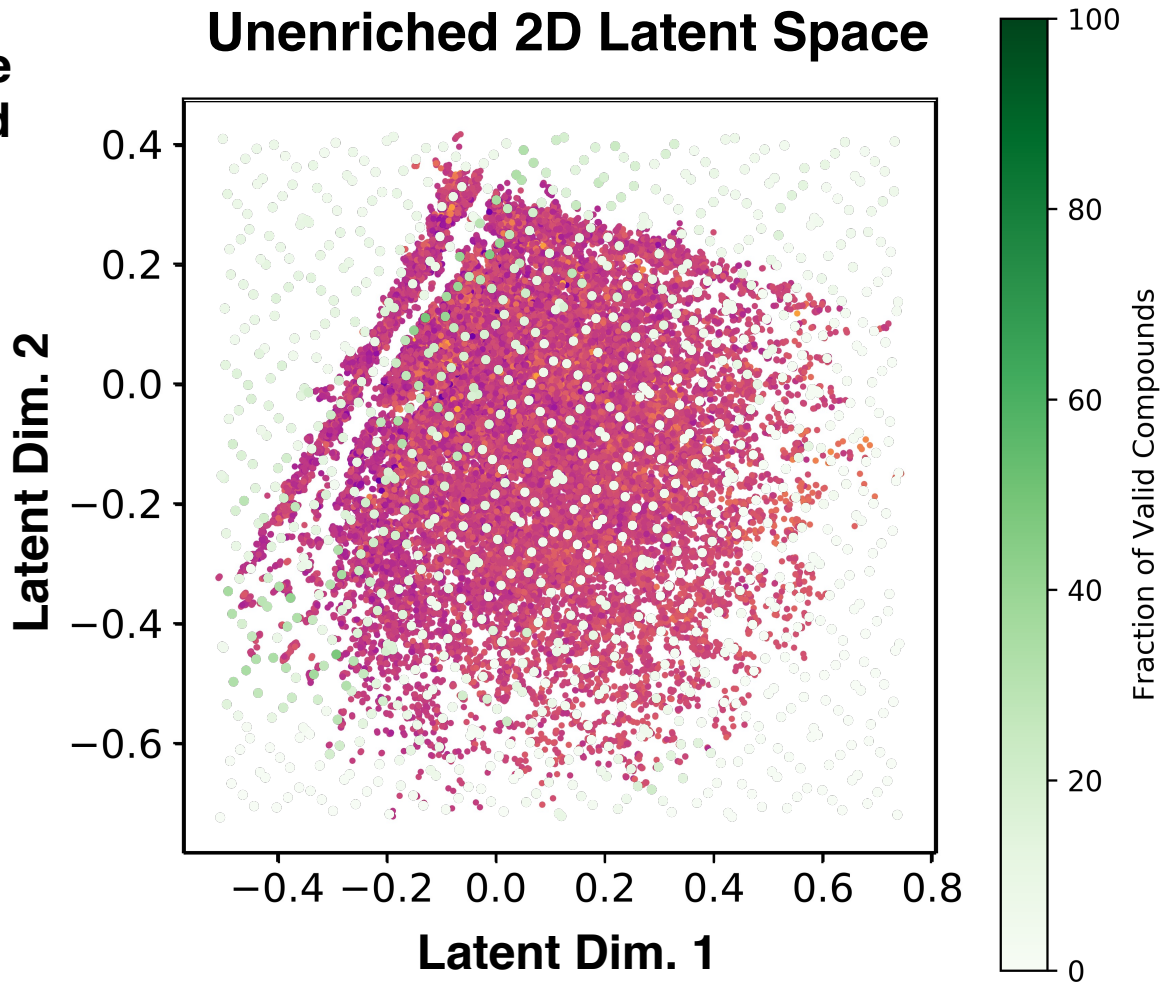
Does the Latent Space Contain Real Molecules Outside of the Training Chemistries?

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?

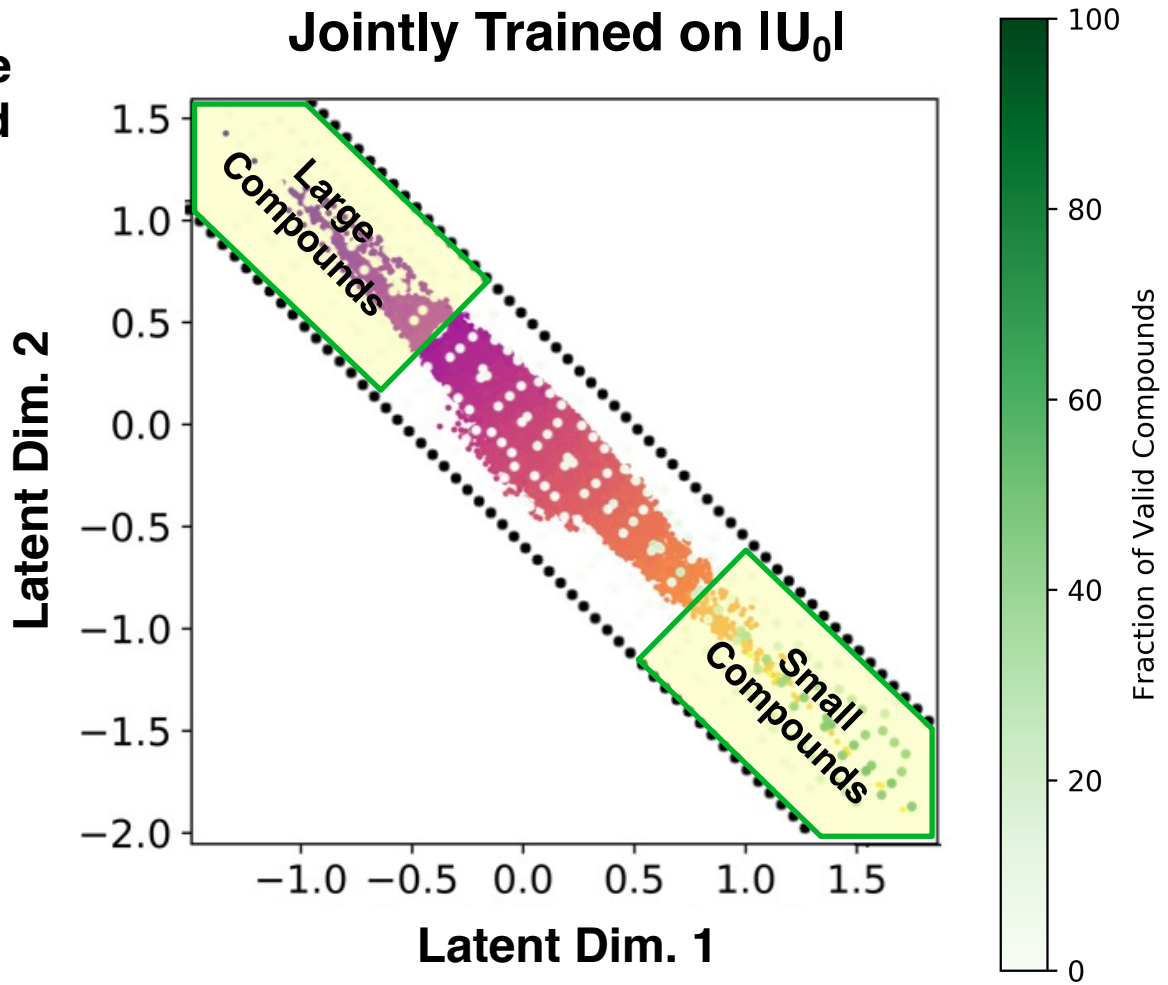
After training only ~4% of the latent space decodes to valid mappings

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

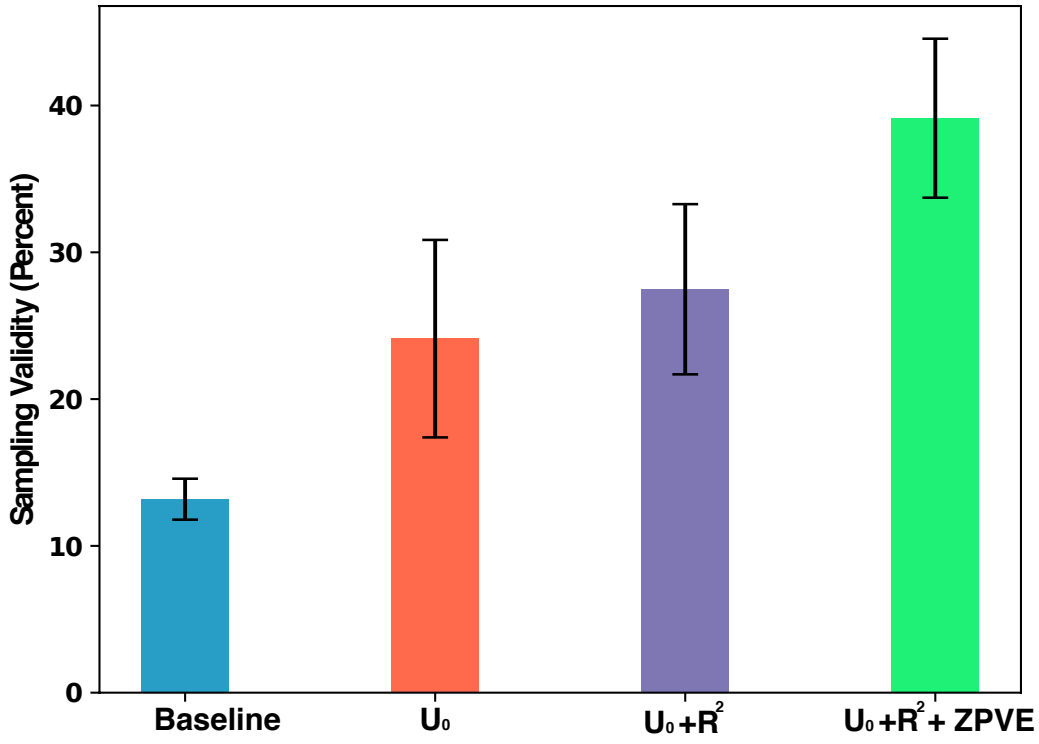
After training only ~4% of the latent space decodes to valid mappings

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

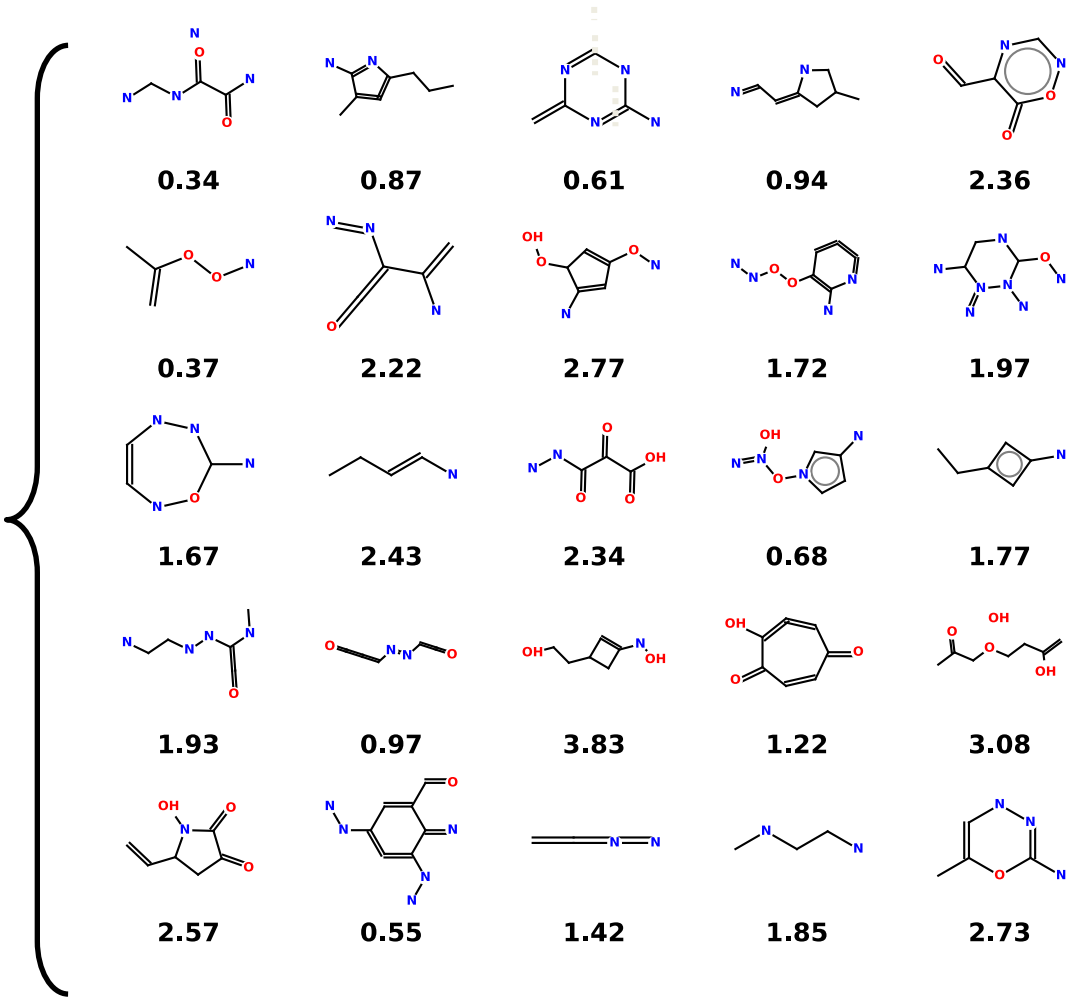


Multi-property training increases the interpretability and validity of structures found in the latent space

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.

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.

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) ⟶ $T_{g,exp} - T_{g,model}$

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.

Example: Ingestion

Preliminary Imports and Data Ingestion:

```
# library imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
```

```
### 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))
```

```
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: Exploring the Data

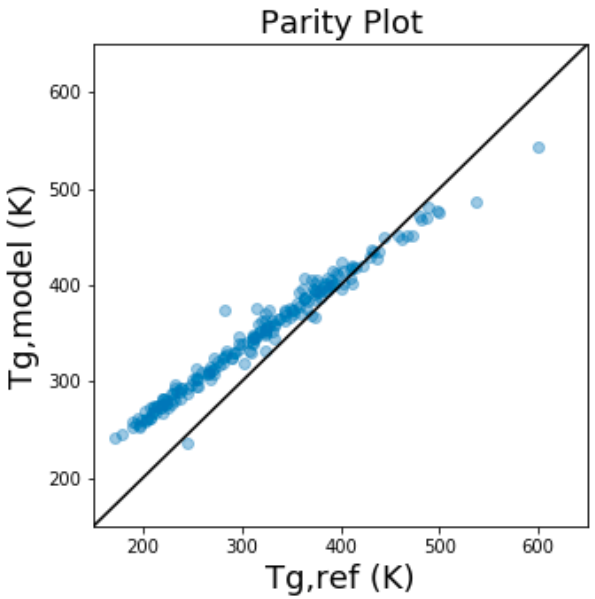
Overview of Supplied Data:

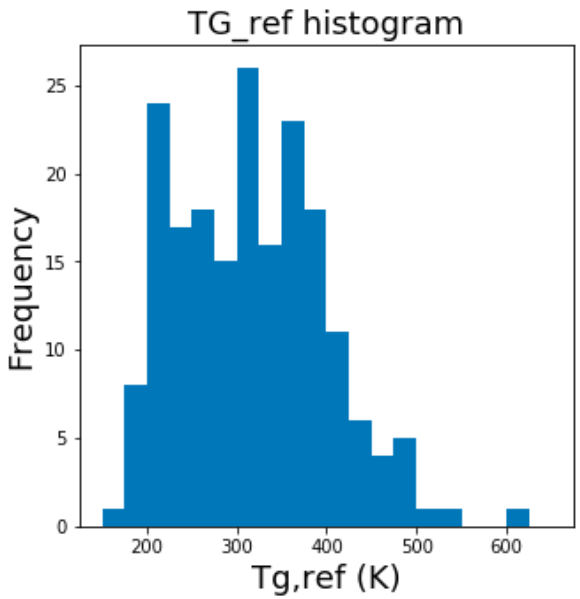
```
# Compute and print summary statistics for TG_model compared with TG_ref
ae = np.absolute(test_data["TG_model"]-test_data["TG_ref"])
print('Mean AE:   {:< 4.2f}'.format(np.mean(ae)))
print('Mean %AE:  {:< 4.2f}'.format(np.mean(ae/test_data["TG_ref"]*100.0)))
print('StdAE:     {:< 4.2f}'.format(np.std(ae)))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

```
Mean AE:    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,verbose=1)
single_model.fit(train_prints,train_data["TG_ref"])
```

If you've never used scikit-learn before, the basic syntax is that each model is programmed as a class, so you initialize an instance of the class (`constructor(hyperparameters)`), then call the `class.fit()` method to actually do the training.

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)))
```

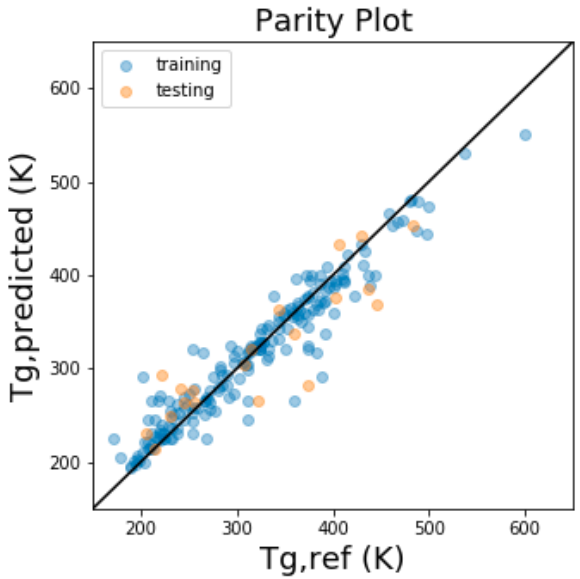
```
Mean AE:    31.34
Mean %AE:   9.84
StdAE:      24.87
Median AE:  24.33
```

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 to poorly. Tg is a hard property to learn, and we've given a relatively generic feature 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,verbose=1)
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.

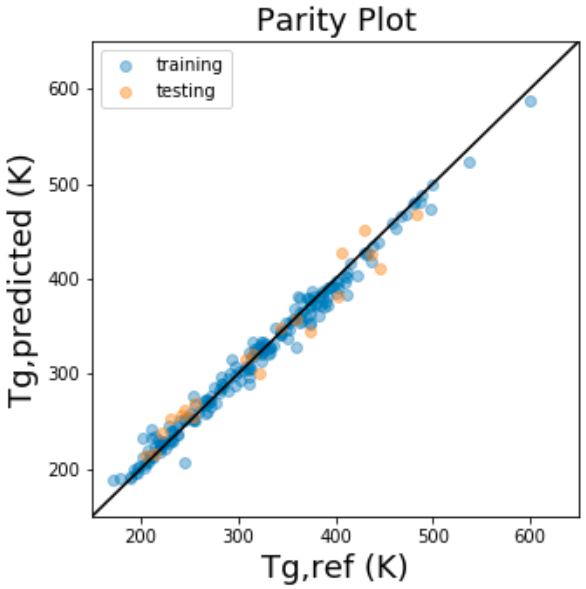
Example: Difference Model

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



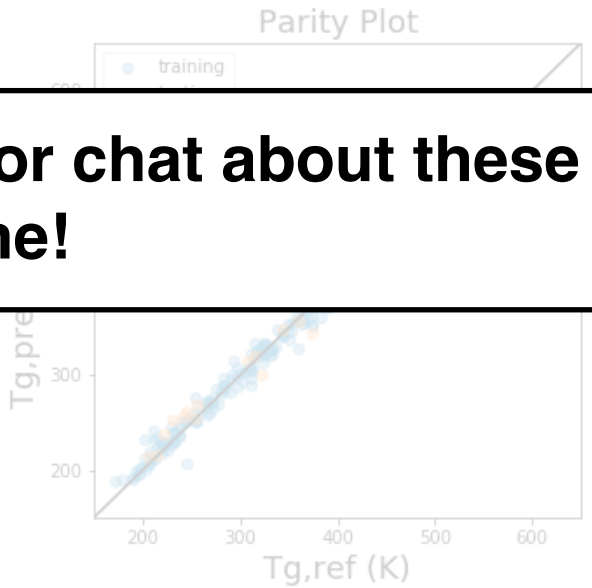
Example: Difference Model

Evaluate the Difference Model Performance:

```
# Parity plot of ref vs predicted Tg in training data and testing data
plt.figure(figsize=(5,5))
plt.p
plt.s
plt.s
plt.x
plt.y
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.

Happy to answer questions or chat about these topics offline!



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