Handling Distribution Shift in Tire Design

Antoine de Mathelin[1,2] **Francois Deheeger**[1] **Mathilde Mougeot**[2,3] **Nicolas Vayatis**[2]
[1]Manufacture Française des Pneumatiques Michelin, Clermont-Ferrand, France
[2]Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, Gif-sur-Yvette, France
[3] ENSIIE, Évry-Courcouronnes, France
antoine.de-mathelin-de-papigny@michelin.com

Abstract

The recent success of machine learning methods in the industrial sector open new perspectives for the design of innovative products. However, these promising results are often challenged when it comes to industrial model deployment. Indeed, it frequently appears that the performance of the model is degraded when used on application data due to the distribution shift between the training and the targeted data. This issue is even more critical for model dedicated to the research of innovative designs as the model is mainly used on unseen regions of the design space. In this work, we present, on a real application of tire design, how distribution shifts impact the model performance and what can be expected from several domain adaptation methods. In an objective of industrial model deployment, we conduct this benchmark with the use of unsupervised evaluation metrics that considerably help the model selection.

1 Introduction

The design of complex industrial products such as tires requires a time consuming and expensive process. Designers need to explore for multiple design choices to find novel products with desired properties. In this context, the use of machine learning models able to learn the relationship between the design of a product and its properties can substantively improve the process [25].

Recent works in design space exploration show that machine learning methods can efficiently learn the mapping between a product components and its properties [3, 11, 12, 16]. However, it has been observed that the learned models are somehow limited to the domain defined by the training data and provide degraded performances on unseen regions of the design space [13]. These observations have been confirmed in our experiments (cf Section 2.1). This is an important drawback as designers aim to explore for innovative designs to find new products with competitive properties.

To tackle this issue, one solution is to correct the shift between the training distribution and the distribution on which the model is applied. To this end, the learner can use domain adaptation methods which either transform the feature space to find more robust representations [8, 19, 17, 26] or reweight the training instances to correct the sample bias [5, 7, 10, 14, 21]. Both strategies have been applied for design space exploration with promising results [1, 13, 15]. However, many questions still need to be addressed before deploying such methods. Because of the huge number of existing methods, practitioners are in particular wondering how to make a relevant choice between them and how to select the right hyper-parameters? They are also wondering what benefit a few labels can bring in the target domain?

In this paper we tackle these questions by studying on a real scenario of technological drift in tire design, the impact of several domain adaptation methods on the target score. We also explore the influence of several unsupervised metrics on the choice of method and hyper-parameters. The organization of the paper is as follows: first we present the issue of distribution shift in tire design and

Workshop on Distribution Shifts, 35th Conference on Neural Information Processing Systems (NeurIPS 2021).

the impact of this shift on the model performances. We then present the metrics that are considered to make the model selection. We finally present the results obtained for real tire design data with the main domain adaptation methods provided in the ADAPT[1] library [6].

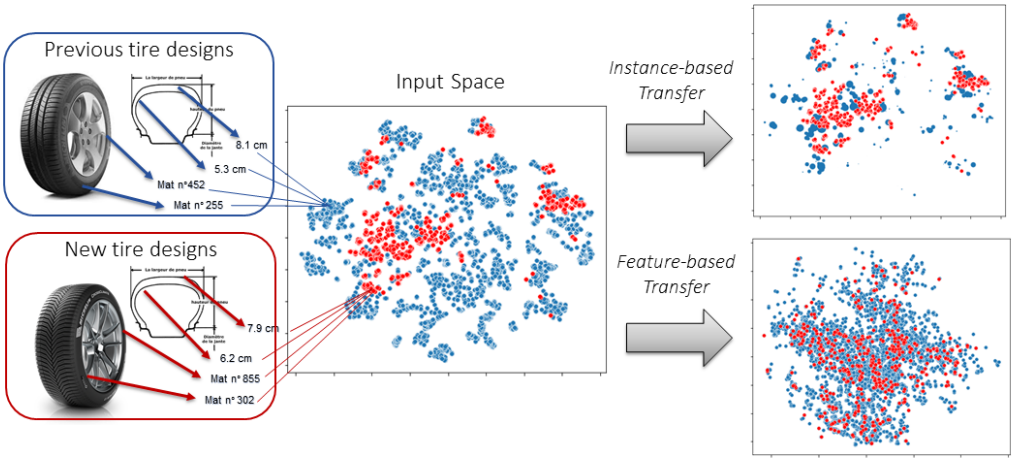

Figure 1: Domain adaptation for tire design: to correct the shift that exists between already deployed tires and potential innovative designs, instance-based or feature-based strategies can be considered.

2 Motivation and Evaluation metrics

2.1 Motivation

We consider the problem of tire performance prediction where the learner has access to a set of *source* labeled tires data $\mathcal{S} = \{(x_1, y_1), ..., (x_m, y_m)\} \in \mathbb{R}^p \times \mathbb{R}$ where each x_i is a vector of size p corresponding to the features of one particular tire's design (i.e. the height and width of the different rubber layers, the materials properties etc...) and y_i is one recorded performance of interest. $\mathcal{S}$ is referred as the source data set which contains the design of already deployed tires. The learner has also access to an unlabeled set of tires $\mathcal{T} = \{x'_1, ..., x'_n\} \in \mathbb{R}^p$ which corresponds to possible new tire's designs. The goal is to find the design x' in $\mathcal{T}$ with the best corresponding performance y'.

The performance y' corresponding to each $x' \in \mathcal{T}$ can be approximated by using a machine learning model $f : \mathbb{R}^p \to \mathbb{R}$. The standard approach consists to train f using only the data set $\mathcal{S}$. We perform this experiment on a set of real tire designs. We first split $\mathcal{S}$ in a train and test sets and fit f on the first set. We then compute the prediction errors on the source train and test sets as well as on the target set $\mathcal{T}$. The results are reported on Figure 2.B and 2.C, we can observe that the errors provided by f on the target set are far above the validation error computed on the source test set. We observe, besides, on Figure 2.A, a clear distribution shift in the input space between $\mathcal{S}$ and $\mathcal{T}$. This shift certainly causes the degraded performances of the model on the target set.

Facing this issue, one solution would be to use a domain adaptation method to correct the shift in the input space. However, as in practice no or only a few labels are available for the data from $\mathcal{T}$, finding the right method to use is not straightforward. Furthermore, many domain adaptation methods are very sensitive to negative transfer [20] and require a careful model selection. For these reasons, we introduce, in the next section, evaluations metrics that will be considered to help the model selection.

2.2 Evaluation Metrics

When dealing with a domain adaptation problem, two main scenarios can be considered: the supervised and unsupervised domain adaptation. In the first one, the learner has access to a small

[1] `https://adapt-python.github.io/adapt/`

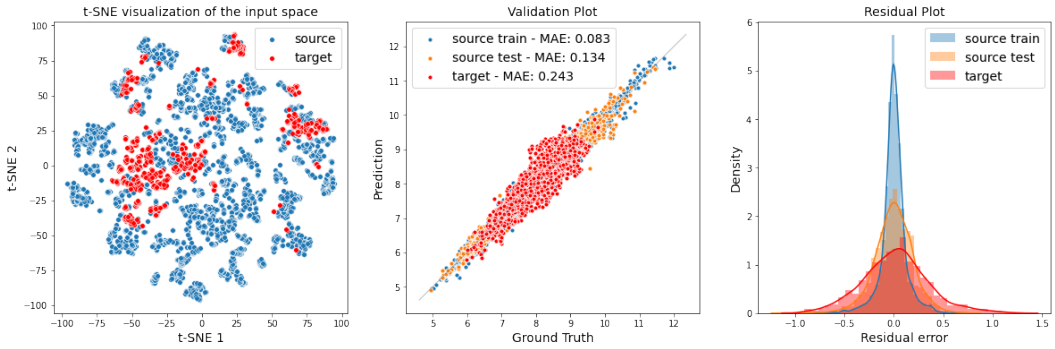

Figure 2: From the left to the right: t-SNE visualization [24] of the input space, validation plot and distribution of the errors for a model trained on source data only.

sample of labeled target data and can then estimate a target score. In the second scenario, no label are available on the target domain. Thus, to help selecting an adequate model, one can only consider unsupervised metrics. In this work, we consider the metrics below, some of them have already been used for hyper-parameter selection as the J-score and the Reverse Validation, others are measures of divergence between distributions:

- **J-score (js)**: is proposed in the method KLIEP [21] for selecting the bandwidth of the kernel, it is correlated to the KL divergence between source and target distributions.
- **Domain Classifier (dc)**: characterizes the shift between domains by the ability of a classifier to discriminate between data from different domains [2, 8].
- **Frechet Distance (fd)**: is inspired from the FID score [9] used for GANs to compare the real distribution to the generaed distribution.
- **Linear Discrepancy (ls)**: is one of the first unsupervised metric used for domain adaptation, it can be computed for linear classifiers using linear algebra [14].
- **Reverse Validation (rev)**: is computed as a source error by inversing the role of the source and target domains, this requires pseudo target labels which are computed using the domain adaptation model [27, 8]

According to [8] a good feature representation is where the source and target distributions are matching and the source task can be learned. Considering only the source error for model selection is not enough as we have shown in Section 2.1 but considering only unsupervised metric may also be inefficient. Indeed, we could produce a representation where all data are regrouped in one localization, thus, the two distributions will match but it will not be possible to learn the task anymore. Thus, to take into account this two objectives, we would like to consider, for model evaluation, a combination of the source error and one unsupervised metric. However, knowing the importance to give to the source error relatively to the metric is difficult. We then propose the following evaluation process: first training several models using the domain adaptation method and then evaluate the source validation error and the unsupervised metric for each run. The source errors and the metrics are then rescaled with standard scaling. An evaluation score is then given by summing them together. We thus give the same importance to the learning of the source task and the matching of distributions.

3 Experiments

3.1 Setup

We perform the experiments on a tire design data set composed of around 7500 tire design vectors of size 470 describing the features of the tire designs. In this data set, we select all data belonging to a same tire line (around 1500 data) which act as the set of potential new tire's designs $\mathcal{T}$. The remaining 6000 data define the set of previous tire's designs $\mathcal{S}$.

We consider the following domain adaptation methods available in the ADAPT library [6]: the two unsupervised instance-based methods KMM [10] and KLIEP [21] and the four unsupervised feature-based methods CORAL [22], DeepCORAL [23], DANN [8] and mSDA [4]. We also consider the supervised method TrAdaBoostR2 [18].

We perform model selection using the metrics describe in Section 2.2. For each domain adaptation method, several models are trained with different set of hyper-parameters and random seed. For each trained model the evaluation metrics are recorded along with a source validation error. The evaluation scores described in Section 2.2 are then computed. We finally select the model which returns the smallest score. The list of hyper-parameters considered for each method as well as the networks architecture and optimization parameters are presented in Appendix A.

3.2 Results and Discussion

The results for the unsupervised experiments are reported in Table 1. We compare the target MAE obtained using the selected model according to each of the unsupervised metrics. The average target MAE (avg tgt) over all trained models is reported along with the MAE of the best model (min tgt).

Table 1: Target MAE across different methods and evaluation metrics. The values below "src+rev" correspond to the target MAEs of the models selected according to the scores computed with the combination of the "rev" metric and the source validation error of each method (cf Section 2.2).

method	src+rev	src+fd	src+ld	src+js	src+dc	avg tgt	min tgt
Src-Only	0.243	0.243	0.243	0.243	0.243	0.243	0.243
KMM	0.236	**0.239**	**0.224**	**0.236**	0.233	**0.235**	**0.217**
KLIEP	**0.223**	0.244	0.228	0.633	**0.228**	0.350	0.223
mSDA	0.349	0.451	0.410	0.559	0.684	0.486	0.296
CORAL	0.254	0.314	0.232	0.266	0.229	0.344	0.221
DeepCORAL	0.577	0.629	0.606	0.539	0.608	0.570	0.406
DANN	0.228	0.331	0.331	0.276	0.642	0.391	0.222

Table 1 first shows that, for any metric, the MAE obtained by selecting the model with the smallest score is most of the time below the average target error (avg tgt). This means that using an unsupervised metric to select the model is in general improving the performances one can expect from domain adaptation against using a model with arbitrary parameters. We observe in particular that the two metrics Reverse Validation (rev) and Linear Discrepancy (ld) help to select better model than other metrics on our tire data set. We finally observe that the two instance-based methods KMM and KLIEP provide the best scores.

We then visualize the impact of adding labels in the target domain. We now select the score according to a validation error obtained on 20% of the labeled target, the 80% others are used during training along with sources. In this category, TrAdaBoostR2 specifically exploits the labeled target data. In this experiment, this last method improves substantively the model performances (cf Table 2).

Table 2: Target MAEs of the models selected with target validation scores

method	5 tgt	10 tgt	15 tgt	20 tgt	30 tgt	50 tgt
No Adapt	0.226	0.212	0.223	0.215	0.219	0.212
KMM	0.226	0.255	0.230	0.197	0.204	0.191
KLIEP	0.229	0.241	0.246	0.219	0.208	0.183
TrAdaBoostR2	**0.216**	**0.188**	**0.198**	**0.176**	**0.191**	**0.176**

4 Conclusion

In this work, we address the challenge of distribution shift in tire design. We show, in particular, that instance-based domain adaptation methods are useful to improve the model performances in both supervised and unsupervised scenarios. We conduct our work in an industrial model deployment perspective and propose, for this purpose, an unsupervised evaluation process to make accurate

model selection. We are convinced that our study will give the practitioners more confidence in using domain adaptation for industrial design.

Acknowledgments

Part of this research was funded by Manufacture Française des Pneumatiques Michelin who provided the tire data set and the Industrial Data Analytics and Machine Learning chair of Centre Borelli from ENS Paris Saclay.

References

[1] *Using Bayesian Optimization With Knowledge Transfer for High Computational Cost Design: A Case Study in Photovoltaics*, volume Volume 2A: 45th Design Automation Conference of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 08 2019. V02AT03A015.

[2] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 137–144. MIT Press, 2007.

[3] Gerd Bramerdorfer and Alexandru-Ciprian Zăvoianu. Surrogate-based multi-objective optimization of electrical machine designs facilitating tolerance analysis. *IEEE Transactions on Magnetics*, 53(8):1–11, 2017.

[4] Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning*, ICML'12, page 1627–1634, Madison, WI, USA, 2012. Omnipress.

[5] Corinna Cortes and Mehryar Mohri. Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519, 2014.

[6] Antoine de Mathelin, François Deheeger, Guillaume Richard, Mathilde Mougeot, and Nicolas Vayatis. Adapt: Awesome domain adaptation python toolbox. *arXiv preprint arXiv:2107.03049*, 2021.

[7] Antoine de Mathelin, Guillaume Richard, Mathilde Mougeot, and Nicolas Vayatis. Adversarial weighting for domain adaptation in regression. *arXiv preprint arXiv:2006.08251*, 2020.

[8] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, January 2016.

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6629–6640, 2017.

[10] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex J. Smola. Correcting sample selection bias by unlabeled data. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 601–608. MIT Press, 2007.

[11] Engin Ïpek, Sally A McKee, Rich Caruana, Bronis R de Supinski, and Martin Schulz. Efficiently exploring architectural design spaces via predictive modeling. *ACM SIGOPS Operating Systems Review*, 40(5):195–206, 2006.

[12] PJ Joseph, Kapil Vaswani, and Matthew J Thazhuthaveetil. A predictive performance model for superscalar processors. In *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, pages 161–170. IEEE, 2006.

[13] D. Li, S. Wang, S. Yao, Y. Liu, Y. Cheng, and X. Sun. Efficient design space exploration by knowledge transfer. In *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 1–10, Oct 2016.

[14] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009.

[15] A. T. W. Min, R. Sagarna, A. Gupta, Y. Ong, and C. K. Goh. Knowledge transfer through machine learning in aircraft design. *IEEE Computational Intelligence Magazine*, 12(4):48–60, 2017.

[16] B. Ozisikyilmaz, G. Memik, and A. Choudhary. Machine learning models to predict performance of computer system design alternatives. In *2008 37th International Conference on Parallel Processing*, pages 495–502, Sep. 2008.

[17] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

[18] David Pardoe and Peter Stone. Boosting for regression transfer. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, June 2010.

[19] Guillaume Richard, Antoine de Mathelin, Georges Hébrail, Mathilde Mougeot, and Nicolas Vayatis. Unsupervised multi-source domain adaptation for regression. In Frank Hutter, Kristian Kersting, Jefrey Lijffijt, and Isabel Valera, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2020, Ghent, Belgium, September 14-18, 2020, Proceedings, Part I*, volume 12457 of *Lecture Notes in Computer Science*, pages 395–411. Springer, 2020.

[20] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*, volume 898, pages 1–4, 2005.

[21] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1433–1440, Red Hook, NY, USA, 2007. Curran Associates Inc.

[22] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[23] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.

[24] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[25] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45, 2016.

[26] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7404–7413, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[27] Erheng Zhong, Wei Fan, Qiang Yang, Olivier Verscheure, and Jiangtao Ren. Cross validation framework to choose amongst models and datasets for transfer learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 547–562. Springer, 2010.

Appendices

A Architectures and hyper-parameters

Table 3: Base estimator used in KMM, KLIEP, CORAL, mSDA and TrAdaBoostR2

Layer type	Layer size	Activation
Fully-connected	100	ReLU
Fully-connected	100	ReLU
Fully-connected	100	ReLU
Fully-connected	10	ReLU
Fully-connected	1	Linear

Table 4: Architecture for the networks used in DANN and DeepCORAL

Network	Layer type	Layer size	Activation
Encoder	Fully-connected	100	ReLU
	Fully-connected	10	ReLU
Task	Fully-connected	100	ReLU
	Fully-connected	10	ReLU
	Fully-connected	1	Linear
Discriminator	Fully-connected	100	ReLU
	Fully-connected	10	ReLU
	Fully-connected	1	Sigmoid

Table 5: Architecture for the networks used in mSDA

Network	Layer type	Layer size	Activation
Encoder	Fully-connected	100	ReLU
	Fully-connected	100	ReLU
	Fully-connected	latent size	ReLU
Decoder	Fully-connected	100	ReLU
	Fully-connected	100	ReLU
	Fully-connected	470	Linear

Table 6: Classifier used for Domain Classifer metric

Layer type	Layer size	Activation
Fully-connected	10	ReLU
Fully-connected	10	ReLU
Fully-connected	1	Sigmoid

Table 7: Hyper-parameters

Method	Hyper-parameters
KMM	sigma = $\{2^i\}_{i\in[\|-5,5\|]}$
KLIEP	max centers = $\{100, 200, 300, 500\}$ (sigma is selected via the LCV procedure)
mSDA	latent size = $\{10, 20, 50, 100\}$, noise level = $\{2^i\}_{i\in[\|-5,5\|]}$
CORAL	lambda = $\{2^i\}_{i\in[\|-10,10\|]}$
DeepCORAL	lambda = $\{2^i\}_{i\in[\|-10,10\|]}$
DANN	lambda = $\{2^i\}_{i\in[\|-5,5\|]}$

B Visualization of the true target error in function of the unsupervised scores

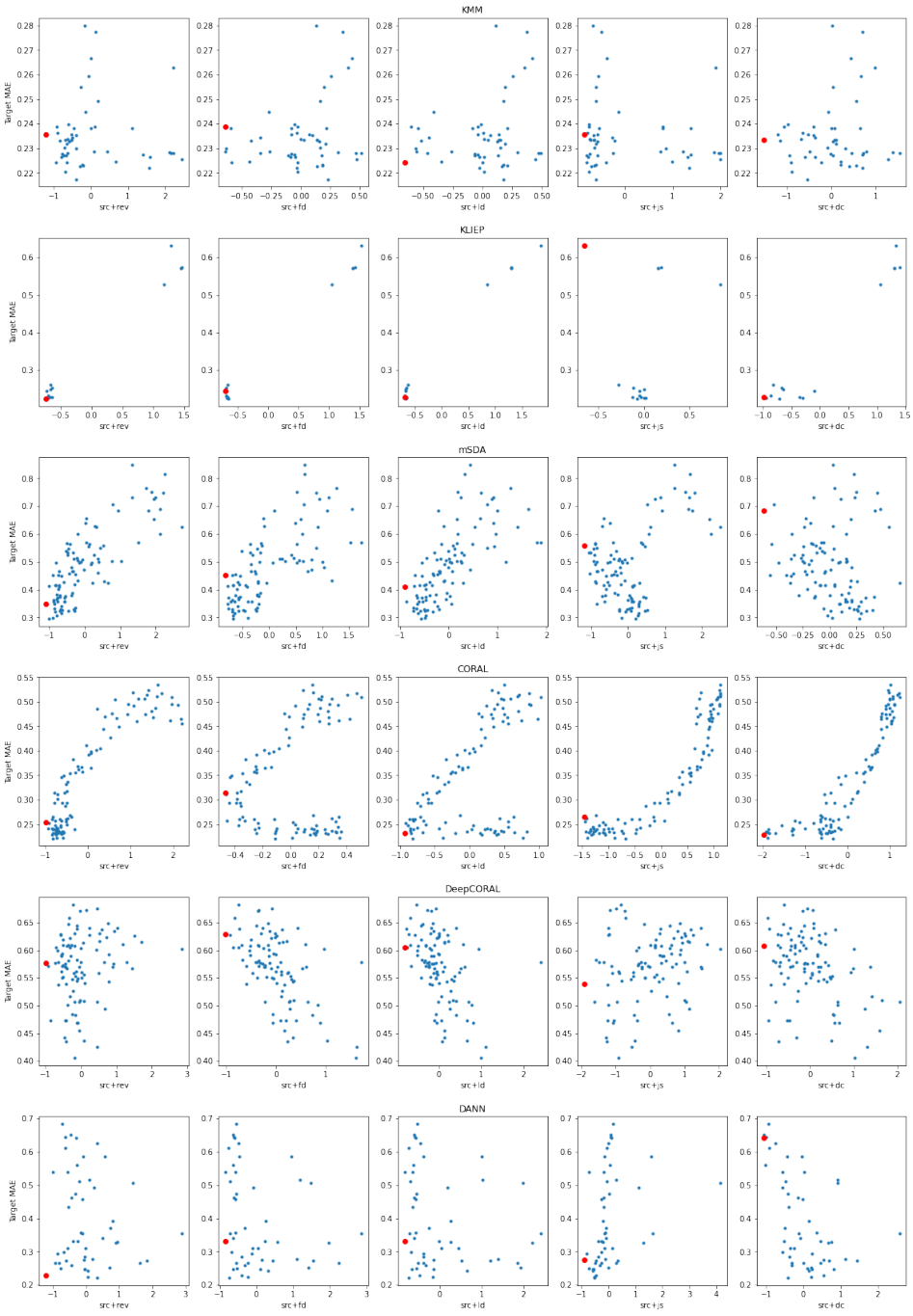

Figure 3: Blue dots show the score evaluated for a model with a particular set of hyper-parameters and random seed. The red dot in each figure is the model selected, i.e. the model with the smallest score. We can see that this model has, in general, a target error below the average target error.