Transformational machine learning: Learning how to learn from many related scientific problems

Ivan Olier, Oghenejokpeme I. Orhobor, Tirtharaj Dash, Andy M. Davis, Larisa N. Soldatova, Joaquin Vanschoren, and Ross D. King

Almost all machine learning (ML) is based on representing examples using intrinsic features. When there are multiple related ML problems (tasks), it is possible to transform these features into extrinsic features by first training ML models on other tasks and allowing them to make predictions for each example of the new task, yielding a novel representation. We call this transformational ML (TML). TML is very closely related to, and synergistic with, transfer learning, multitask learning, and stacking. TML is applicable to improving any nonlinear ML method. We tested TML using the most important classes of nonlinear ML: random forests, gradient boosting machines, support vector machines, k-nearest neighbors, and neural networks. To ensure the generality and robustness of the evaluation, we utilized thousands of ML problems from three scientific domains: drug design, predicting gene expression, and ML algorithm selection. We found that TML significantly improved the predictive performance of all the ML methods in all the domains (4 to 50% average improvements) and that TML features generally outperformed intrinsic features. Use of TML also enhances scientific understanding through explainable ML. In drug design, we found that TML provided insight into drug target specificity, the relationships between drugs, and the relationships between target proteins. TML leads to an ecosystem-based approach to ML, where new tasks, examples, predictions, and so on synergistically interact to improve performance. To contribute to this ecosystem, all our data, code, and our ∼50,000 ML models have been fully annotated with metadata, linked, and openly published using Findability, Accessibility, Interoperability, and Reusability principles (∼100 Gbytes).

M
machine learning (ML) develops computational systems that learn from experience (1–4). ML has a long history of application to science, one of the first ML programs being Meta-Dendral, which used ML to improve the analysis of mass-spectrometric data (5). The importance of ML to science is now widely recognized, and ML is now being applied to almost all areas of science: drug discovery (6), organic synthesis planning (7), materials science (8), medicine (9), and so on.

Most ML represents examples using tuples of attributes, i.e., the data can be put into a single table, with the examples as rows and the attributes as columns (1–4). Attributes are features of examples which are believed to be important. Currently, such features are almost always intrinsic properties. For example, if one wished to learn about the pharmacological activity of a drug, then properties of its molecular structure would be useful attributes. Typically, one attribute is singled out for prediction, and the other attributes contribute information to make this prediction. If the predicted attribute is categorical then the problem is a discrimination/classification task, and if the attribute is a real number then the problem is a regression one. Here we focus on regression.

In cases where there are multiple related ML problems (tasks) it is possible to use extrinsic features: predictions made about examples by ML models learned on other tasks. We call this transformational ML (TML). TML transforms a representation based on intrinsic attributes of examples to an extrinsic representation based on the predictions of previously learned models. As we will discuss, TML is very closely related to and synergistic with stacking, multitask learning (MTL), and transfer learning (TL). It enables the utilization of knowledge previously learned from related tasks, rather than learning each new model from scratch. TML is thus a metalearning idea that is applicable to enhancing any nonlinear ML method. It is particularly well suited when there exist many related small learning tasks.

To intuitively explain this idea, we take as an illustrative example the problem of learning to recognize multiple animal species (Fig. 1A). If there are many types of animals, with new ones expected to be added, then it would be reasonable to learn classifiers for each species, rather than learning a single large classifier. The standard (baseline) ML approach to such a learning task would be to use intrinsic attributes (e.g., size or presence of fur) to learn these prediction models. The TML approach would be to use extrinsic attributes (e.g., the turns learned by the other species as they were learned) to help the construction of a new model that learns the classification of a new species.
approach is to first learn prediction models for all known species in the standard way and then to use the predictions from these learned baseline models to represent all animals, i.e., by their “catness,” “rabbitness,” or “horseness,” and to train a (meta) ML model (Fig. 1A) to make predictions using this representation. TML is applicable to any domain where ML tasks share a common set of intrinsic features and related target variables, which is commonly the case in scientific domains, e.g., drug design where targets (proteins) can be related, as can drugs (Fig. 1B). The underpinning justification for TML is utilization of prior knowledge about the regularity of the world encoded in the previously learned prediction models.

More formally, the input to TML is a set of previously learned prediction models and a new learning task with labeled examples. TML is performed in two stages. First, the examples from the new learning task are applied to the prediction models and the predictions of the models used to generate the transformed representation. Then, the transformed representation is used to learn a prediction model for the new task (Fig. 1). Consider a set of $n$ tasks (learning problems) $T_i, i = 1, \ldots, n$, each represented by a common set of $p$ attributes $X_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$, and a unique prediction attribute $y_i$. On each task we train a model using a baseline ML method $A_i$, yielding $n$ models $A_i = A(X_i) \approx y_i$. We then apply an ML method $\Phi$ (possibly different from $A$) to predict a new target $y_{new}$ for a new task $T_{new}$ by using the $n$ previously trained models $A_i$ to generate $n$ latent features $y_i$, and learning the relationship from the latent features to the new target $y_{new}$: $\Phi(X_{new}) = \Phi(A_1 \ (X_{new}), A_2 \ (X_{new}), \ldots, A_n \ (X_{new})) = \Phi(y_1, y_2, \ldots, y_n) \approx y_{new}$.

In the case of QSAR (quantitative structure-activity relationships) predictions, a common step in early-phase drug discovery (23, 24), the task $T_i$ is as follows: given a target (usually a protein) and a set of chemical compounds (small molecules) with associated activities (e.g., inhibition of the target protein), learn a predictive mapping from molecular representation to activity: $X_i$ is a set of drug descriptors with known activity $y_i$ (Fig. 1B). Baseline ML methods (e.g., random forest and $k$-nearest neighbor [k-NN]) are first applied to each QSAR prediction task $T_i$ yielding prediction models $A_i \rightarrow activity$, for example $A_1$, $A_2$, $\ldots, A_n$ (Fig. 1C).

In the TML approach, for a new QSAR task we apply an ML method $\Phi$ (which could be different from any of previously used $A_i$, or one previously used) to...
make a new prediction by using the \( n \) previously trained models \( A(X) \). The attributes are now the predictions from baseline QSAR models \( A_i \) (of the form \( \Phi(0.2, 0.3, \ldots) \to 0.9 \); see Fig. 1C).

TML has very close similarities to other ML approaches. However, the specific TML concept does not seem to have been previously identified or systematically evaluated. TML has very close similarities to MTL (10). In MTL related problems (tasks) are learned simultaneously with the aim of exploiting similarities between the problems to improve performance. MTL has been successful in many scientific applications (e.g., refs. 11 and 12). MTL is defined as “an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias. It does this by learning tasks in parallel while using a shared representation; what is learned for each task can help other tasks be learned better” (10). MTL starts, as does TML, with a set of \( n \) task (learning problems) \( T_i, i = 1 \ldots n \), each represented by a common set of \( p \) attributes \( X_i = (x_1, x_2, \ldots, x_p) \), and a unique prediction attribute \( y_i \). MTL aims to improve the learning of a model for \( T_i \) using ML method \( A \) by learning in parallel all the \( n \) models \( A_i = A(X_i) = y_i \). There are two main differences between MTL and TML: TML typically learns the tasks in parallel, while TML typically learns tasks sequentially, and in TML information is shared between tasks using the data representation, while MTL uses a single model.

TML is also very closely related to TL (13), where information is transferred from a specific source problem to a specific target problem. The idea of TL is to extract knowledge from one or more source domains and to reuse this knowledge in a target domain where data are scarce, with the aim of building better-performing learning models in the target domain. Lin and Jung (14) define TL as “given a source domain \( D_S \) and learning task \( T_S \), and a target domain \( D_T \) and learning task \( T_T \), TL aims to help improve the learning of the target predictive function \( f(\bullet) \) in \( D_T \) using the knowledge in \( D_S \) and \( T_S \), where \( D_S \neq D_T \) or \( T_S \neq T_T \)” (14). This definition of TL is very general but typically TL differs from TML in that just one source task is learned, while TML requires many source tasks. TL has previously been successfully applied in drug design with several prospective applications demonstrating the usefulness of this ML approach (15).

TML resembles MTL in using a single joint representation and TL and metalearning in transferring task information. However, in TML instead of using a predefined similarity measure or another criterion to preselect a set of similar tasks the different tasks are projected into a joint numeric representation (embedding), and then any ML can be applied to this new transformed representation to learn how to make accurate predictions for a specific problem.

TML also has very close similarities to stacking (16, 17), a form of ensemble ML. In ensemble ML multiple learning methods are combined to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. In stacking multiple baseline models are first learned, then a metamodel is learned using the outputs of the baseline level model. Stacking starts with a single task \( T_r \), represented by a set of \( p \) attributes \( X_r = (x_1, x_2, \ldots, x_p) \), and a unique prediction attribute \( y_r \). We then train \( m \) baseline models using \( m \) baseline ML methods \( A_j, j = 1 \ldots m \). \( A_j(X) = y_j \). We then apply a ML method \( \Phi \) (possibly different from any \( A_j \)) to learn the relationship between the latent features \( y_r \) and \( y \). The main difference between TML and stacking is that TML learns across a large set of tasks \( T_i, i = 1 \ldots n \), each containing potentially different examples, while in stacking different baseline models are typically trained on the same task; e.g., one might stack together random-forest and neural-network predictors for a specific problem. In contrast, in TML the models are not trained on the same task and could simply be a set of pretrained models.

Within the field of drug design TML also closely resembles the idea of using ML models to predict affinity fingerprints (18). Similarly, in natural language processing, Stubb et al. (19) have successfully used a MTL/TL approach that resembles TML.

TML also resembles the concept of an inductive database (ID) (20) in its focus on multiple models. An ID is a general-purpose database in which both the data and ML models can be represented, retrieved, and manipulated. TML is similar in its focus in considering ML models to objects of interest outside of their initial purpose. It differs in being directly focused on a specific method of using models to aid prediction.

TML is applicable to improving any nonlinear ML method. To evaluate TML we selected five ML methods that exemplify the main families of nonlinear ML methods (1–4): random forests (RF) (21), gradient-boosting machines (XGB) (22), support-vector machines (SVMs) (23), k-NN (3), and neural networks (NN) (3, 4). To ensure the generality and robustness of the evaluation we utilized thousands of ML problems from three important scientific problems: drug discovery (QSAR learning), predicting human gene expression (across different tissue types and drug treatments), and metamachine learning (predicting how well ML methods will perform on problems). For each ML method, and each problem area, we compared TML vs. baseline (standard) ML. We investigated two forms of prediction improvement: strong improvement, where predictions made using the new TML features outperform those based on the baseline (intrinsic) features \( \Phi(X_{TML}) \) vs. \( \Phi(X_{baseline}) \) and combined improvement, where the new TML features improve performance through augmenting the baseline features \( \Phi(X_{TML} \text{ plus } X_{baseline}) \) vs. \( \Phi(X_{baseline}) \). To augment the TML predictions we used the simplest possible form of stacking: combining the predicted outputs. We found that TML significantly improved the average predictive performance of all methods in all three domains (from 4 to 50%), i.e., models trained on the novel extrinsic features generally outperformed those trained on the intrinsic ones (Table 1).

Almost every form of statistical and ML method has been applied to QSAR learning (24, 25), but no single method has been found to be clearly best. (24, 25). QSAR problems are well-suited to TML as they can be related by having related target proteins (e.g., the problem of inhibiting the enzyme dihydrofolate reductase [DHFR] in Mus musculus [mouse] and Homo sapiens are similar because they have similar ligand binding sites [active centers] (26), and they involve the same or chemically related molecules (26–28). To evaluate TML for QSAR learning we utilized 2,219 QSAR problems (24, 25). The baseline (intrinsic) QSAR representation was a 1,024-bit molecular fingerprint representation, which has previously been shown to be effective (25). For each ML method (RF, SVM, k-NN, and NN) we generated the TML extrinsic attributes by predicting compound activities using the previously learned ML models (see SI Appendix, QSAR Learning) then learned the TML QSAR models using the same ML method. Use of TML outperformed baseline ML for all methods. The results are given in Table 1. We found that the best overall results were for stacked TML XGB, which achieved a 7% overall improvement over baseline XGB, followed by TML NN. It is noteworthy that these datasets have been extensively studied (18 learning methods and 6 molecular representations (25)), and TML significantly outperformed the best previous results.

For our second problem domain we used the Library of Integrated Network-based Cellular Signatures data (LINCS) (29), which describes the measured expression levels of 978 landmark human genes under 118,050 experimental conditions. We cast the ML task as learning a model for each gene able to
predic... level given experimental conditions (cell type, drug, and dosage) (see SI Appendix, Gene Expression Learning). The problem domain is suitable for TML as there are relationships between genes (homologies, common signaling pathways, etc.) and between experimental conditions (drug similarity, etc.) that can be exploited to improve performance. Using the same methodology as on the QSAR problem we performed a comparative assessment of RF, SVMs, k-NN, and NNs on the original intrinsic representation and the TML representation. The results for the problem are given in Table 1. Use of TML outperformed baseline ML for all methods. We found that the best overall result was for TML RF with a 4% overall improvement over baseline RF, followed by TML XGB and TML SVM.

Our third evaluation problem area was from ML, where a fundamental problem is to select the best ML method to use on a new learning task. An effective approach to this task is to use ML to solve this problem, which is a kind of metalearning (30). The ML task is to learn a metamodel to predict the performance of an ML method (given an exact configuration) on a new task, given the characteristics of the training data (e.g., statistics of the training data distribution). The problem domain is suitable for TML as ML tasks can be related by having similar data distributions and data properties (e.g., missing values) or by containing data being generated by similar processes. From OpenML (31) we took a set of 10,840 evaluations on 351 tasks (datasets) and 53 ML methods, which produced 351 learning tasks (SI Appendix, Meta-Learning for Machine Learning). The results for the problem are given in Table 1. Use of TML outperformed baseline ML for all methods. We found that the best overall result was TML stacking RF, with a 50% improvement over baseline XGB and a 4% improvement over RF. A similar level of improvement was found for TML XGB over baseline XGB, with TML SVM and TML NN producing the best SVM and NN results. For k-NN stacking TML performed best. The percentage improvement with TML was much greater with the other tasks. This may be due to the fact that the original (intrinsic) features are rather weak descriptors of the training datasets, while the TML features encode much more implicit information about how algorithms behave on different tasks. In addition, measuring predictive performance has lower empirical noise than the other problem areas.

An increasingly important branch of ML is explainable AI, for in many applications (e.g., medical or financial) there is a necessity to make predictions understandable (32). In science, explainable ML predictions lead to new scientific insights. The understandability of ML models depends on model simplicity...
The seven non-DHFR targets seem to have little in common with DHFR; however, digging deeper provides biological insight. It is interesting that shared requirement for a donor/acceptor interaction to glutamic acid. It is also noteworthy that all kinase inhibitors hinge binders also share a similar donor acceptor interaction, which might also explain the kinase model.

Table 2. Top 10 models used to predict DHFR activity

| Order | Target ID   | Weight | Name                                | Species               |
|-------|-------------|--------|-------------------------------------|-----------------------|
| 1     | CHEMBL2902  | 13.5   | Dihydrofolate reductase              | L. casei              |
| 2     | CHEMBL5372  | 12.2   | Methionyl-tRNA synthetase            | Staphylococcus aureus |
| 3     | CHEMBL3048  | 10.6   | Nitric-oxide synthase, brain         | Rattus norvegicus     |
| 4     | CHEMBL2111414 | 8.6 | Tyrosine-protein kinase ABL          | H. sapiens            |
| 5     | CHEMBL329    | 8.5    | Type-1A angiotensin II receptor      | R. norvegicus         |
| 6     | CHEMBL2014   | 7.4    | Nicotinceptor                       | H. sapiens            |
| 7     | CHEMBL5491   | 7.2    | Serine/threonine-protein kinase WEE1 | H. sapiens            |
| 8     | CHEMBL5441   | 7.2    | Dihydrofolate reductase              | E. coli               |
| 9     | CHEMBL5457   | 6.8    | Dihydrofolate reductase              | M. avium              |
| 10    | CHEMBL075294 | 6.6    | Indoleamine 2,3-dioxygenase 1        | M. musculus           |

The seven non-DHFR targets seem to have little in common with DHFR; however, digging deeper provides biological insight. It is interesting that shared requirement for a donor/acceptor interaction to glutamic acid. It is also noteworthy that all kinase inhibitors hinge binders also share a similar donor acceptor interaction, which might also explain the kinase model.
with three clusters and associated compounds. Although the pharmacology of these compounds is complex, it is known to be based on a nexus of serotonin and dopamine receptor interactions. These interactions are correctly predicted by TML models and used in the clustering (see SI Appendix, Predicted Drug Activities). The pharmacology of compounds is explained by their relative position in the clustering (Fig 2B).

We applied an analogous approach to the bioinformatic problem of estimating the similarity of protein targets (Fig. 2C) (see SI Appendix, Clustering). The standard approach to this task is to use evolutionary distance estimated by sequence comparison. However, what is most important in most problems is not evolutionary distance but the functional similarity of protein active sites. We can estimate this using the accumulated information in the TML QSAR models. For each of the targets we calculated its drug-activity profile: the predicted activity of FDA-approved compounds on the target. As with chemical compounds we argue that this clustering is more informative in drug design than conventional evolutionary distance, as it is based on how the target empirically responds to chemical compounds. One intriguing cluster of proteins (drug targets) identified by the similarity of their QSAR predictive models is shown in Fig. 2C. Although the proteins in the cluster do not share any obvious structural similarity, there is a clear theme to the function of these (mammalian) proteins, related to control of metabolism.

It is instructive to compare TML with the currently most significant form of ML, that of deep neural networks (DNNs) (35). DNN input is typically spatially or sequentially structured, and prior knowledge of structure is encoded in the structure of the network. This learned structure is latent. The success of DNNs is based on their ability to utilize multiple neural network layers, and very large amounts of data, to learn how to map poor input representations (e.g., image pixel values) into rich and effective latent representations. This is achieved through use of a differentiable learning model and end-to-end learning. The ability to improve weak input representations has enabled DNNs to succeed in domains that had previously proved recalcitrant to ML: beating World Champions at games such as Go (36), diagnosing skin cancers better than human specialists (9), and so on. A key lesson from the success of DNNs is therefore to use ML to enhance ML representations—which is precisely what TML

![Diagram of TML](https://example.com/diagram.png)

**Fig. 3.** (A) Baseline TML. Each learning task is illustrated as an oval, models are squares, and learning methods diamonds (for clarity we only show RF). We focus on learning task T1. In (i) baseline learning is used to learn RF models for all the tasks (problems). In (ii) the predictions from the different learned RF models are used to learn a TML model for task T1. In (iii) stacking is used to combine the RF model RF, and TML model TML, to form the output stacking model S. (B) Variants of TML. The figure illustrates three possible variants of TML applied to the same problem P1. (I) Shows TML feature selection. This selection could be done using baseline ML feature selection methods, or based on understanding the semantics of the relationships between the tasks (e.g., in drug design one might wish to consider the homology of related targets), and so on. (II) Illustrates the use of ensemble learning at the TML level. RF, SVM, k-NN, and NN are combined together using stacking. (III) Illustrates second-order (multilevel) TML. (C) ML as an ecosystem. Currently ML tasks are generally seen as to be solved stand-alone, or perhaps in small groups by MTL and TL. The TML-inspired view is that learning is cumulative and never-ending: When learning a new task, one should utilize existing models (knowledge) even if they were learned for different (but related) tasks, and when new problems, methods, models, and examples appear these should be used to improve existing models and predictions.
The traditional approach to ML is to view each learning task as a separate problem. This view is starting to change with progress in TML (10), TL (13), life-long learning (37), and so forth. TML leads to an even broader vision of ML as an ecosystem (Fig. 3C). In this ecosystem, learning tasks, learning examples, ML methods, ML predictions, meta-ML methods, and so on all interact synergistically to produce enhanced performance and understanding over all tasks in the ecosystem. If more knowledge is added through TML, second-order transformations may be relearned. TML adds some additional computational costs to learning, but the additional cost of TML is low compared to DNN learning.

We have presented only the most basic form of TML (Fig. 3A). Many other variants with potentially greater performance are possible. For example, it is possible to integrate TML with feature selection and stacking in multiple possible ways (Fig. 3B). Given that TML can often produce better predictions than the original intrinsic representation, it is natural to extend the idea of TML by applying it a second time, i.e., to use the predictions from the transformed representation to form a second-order transformed representation (Fig. 3B). It is of course also possible to combine feature selection, ensemble learning, stacking, TML, second-order TML, and so on.

Within ML there is increasing interest in the automation of ML, and there exist a number of both free and commercial systems that automate the application of ML to new problems (39). For example, Auto-WEKA and Auto-sklearn (39) search through a space of possible ML methods, and hyperparameters, to optimize ML predictions. However, no current ML automation system has discovered a valuable new ML idea such as dropout, stacking, and so on. Although there is increasing amount of research on AI systems designed to automate scientific discovery (40), and these systems are heavily based on ML, little work has been done on applying AI discovery systems to ML. The development of a ML system able to learn important new ML ideas would transform ML and the world.

**Materials and Methods**

To enable reproducibility, all of the thousands of datasets (QSA, LINCS, and Metalearning), the links to the code (TML, RF, XGB, SVM, k-NN, NN), and the −50,000 ML RF (counting all decision trees) models are available under the creative commons license at the Open Science Platform: https://osf.io/vbn5u/. This amounts to ∼100 Gbytes of compressed data. Few ML projects have put online so much reusable data. To maximize its added value we follow the FAIR (Findability, Accessibility, Interoperability, and Reusability) principles for publishing digital objects (41) (see SI Appendix, FAIR Sharing).

**Data Availability.** Datasets, code, and ML models reported in this study have been deposited in Open Science Framework (https://osf.io/vbn5u).

**ACKNOWLEDGMENTS.** The work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, by the Engineering and Physical Sciences Research Council projects Robot Chemist and Action on Cancer, and by the Alan Turing Institute project Spatial Learning: Applications in Structure Based Drug Design. We thank Rafael Mantovani for generating the original meta-learning data used in this study and Ljupco Todorovski for useful discussions.

### References

1. T. M. Mitchell, Machine Learning (McGraw-Hill, 1997).
2. S. J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach (Pearson, 2016).
3. R. O. Duda, P. E. Hart, Pattern Classification (Wiley-Interscience, 2000).
4. C. Bishop, Pattern Recognition and Machine Learning (Springer, 2006).
5. B. Buchanan, G. Sutherland, E. A. Feigenbaum, *Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry* (Stanford University, 1968).
6. G. Schneider, Automating drug discovery. *Nat. Rev. Drug Discov.* 17, 97–113 (2018).
7. M. H. S. Segler, M. Preuss, M. P. Waller, Planning chemical syntheses with deep neural networks and symbolic AI. *Nature* 555, 604–610 (2018).
8. K. T. Butler, D. W. Davies, H. Cartwright, O. Iyaye, A. Walsh, Machine learning for molecular and materials science. *Nature* 559, 547–555 (2018).
9. A. Esteva et al., Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 542, 115–118 (2017).
10. R. Caruana, Multitask learning. *Mach. Learn.* 28, 41–75 (1997).
11. S. Bickel, J. Bogojicja, T. Lengerger, T. Scheffer, “Multi-task learning for HIV therapy screening” in Proceedings of the 25th International Conference on Machine Learning (Association for Computing Machinery, 2008) pp. 56–63.
12. N. Sadawi et al., Multi-task learning with a natural metric for quantitative structure activity relationship learning. *J. Cheminform.* 11, 68 (2019).
13. S. Thrun, L. Pratt, “Learning to learn: Introduction and overview” in Learning to Learn, S. Thrun, L. Pratt, Eds. (Springer, 1998), pp. 3–17.
14. Y.-P. Lin, T.-P. Jung, Improving EEG-based emotion classification using conditional transfer learning. *Front. Hum. Neurosci.* 11, 334 (2017).
15. G. Schneider, D. E. Clark, Automated de novo design – “Are we nearly there yet?” *Angew. Chem. Int. Ed. Engl.* 58, 10792–10803 (2019).
16. D. H. Wolpert, Stacked generalization. *Neural Netw.* 5, 241–259 (1992).
17. E. Spyromitros-Xioufis, G. Tsoumakas, W. Groves, I. Vlahavas, Multi-target regression via input space expansion: Treating targets as inputs. *Mach. Learn.* 104, 55–98 (2016).
18. I. Cortes-Ciriano, C. Škuta, A. Bender, D. Svozil, QSAR-derived affinity fingerprints (part 2): Modeling performance for potency prediction. *J. Cheminform.* 12, 41 (2020).
19. E. Strubell, P. Verga, D. Andor, D. Weiss, A. McCallum, “Linguistically-informed self-attention for semantic role labeling” in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (Association for Computational Linguistics, 2018), pp. 5027–5038.
20. T. Imieliński, H. Mannila, A database perspective on knowledge discovery. *Commun. ACM* 39, 58–64 (1996).
21. L. Breiman, Random forests. *Mach. Learn.* 45, 5–32 (2001).
22. M. J. Baxter, P. B. Bartlett, M. Frean, “Boosting algorithms as gradient descent” in Advances in Neural Information Processing Systems, S. A. Solla, T. K. Leen, K. Müller, Eds. (MIT Press, 1999), vol. 12, pp. 512–518.
23. A. Cortes, V. Vapnik, Support-vector networks. *Mach. Learn.* 20, 273–297 (1995).
24. A. Cherkasov et al., QSAR-modeling: Where have you been? Where are you going to? *J. Med. Chem.* 57, 4977–5010 (2014).
25. I. Olier et al., Meta-QSAR: A large-scale application of meta-learning to drug design and discovery. *Mach. Learn.* 167, 285–311 (2018).
26. A. Goulton et al., The CHEMBL database in 2017. *Nucleic Acids Res.* 45, D945–D954 (2017).
27. M. Weisel, E. Proschak, J. M. Kriegel, G. Schneider, Corner forms function follows: Shape analysis of protein cavities for receptor-based drug design. *Proteomics* 9, 451–459 (2009).
28. H. Oztürk, E. Oztürk, A. Bender, D. Svozil, Distributed representations of proteins using their interacting ligands. *Bioinformatics* 34, 1295–1303 (2018).
29. A. Koleti et al., Data portal for the Library of Integrated Network-based Cellular Signatures (LINCS) program: Integrated access to diverse large-scale cellular perturbation response data. *Nucleic Acids Res.* 46 (D1), D558–D566 (2018).
30. J. Vanschoren, Meta-learning: A survey. arXiv [Preprint] (2018). https://arxiv.org/abs/1810.03548 (Accessed 1 August 2020).
31. J. Vanschoren, J. N. van Rijn, B. Bischl, L. Torgo, OpenML: Networked science in machine learning. *SIGKDD Explor.* 15, 49–60 (2013).

Oiler et al.

Transformational machine learning: Learning how to learn from many related scientific problems.
32. T. Miller, R. Weber, D. Maggazeni, Eds., Proceedings of the IJCAI 2019 Workshop on Explainable Artificial Intelligence (2019). https://sites.google.com/view/xai2019/home. Accessed 1 August 2020.
33. P. Gardenfors, Conceptual Spaces: The Geometry of Thought (MIT Press, 2004).
34. M. A. Boden, Mind as Machine: A History of Cognitive Science (Oxford University Press, 2006).
35. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature 521, 436–444 (2015).
36. D. Silver et al., Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489 (2016).
37. S. Thrun, T. M. Mitchell, Lifelong robot learning. Robot. Auton. Syst. 15, 25–46 (1995).
38. E. T. Jaynes, Probability Theory: The Logic of Science (Cambridge University Press, 2003).
39. F. Hutter, L. Kotthoff, J. Vanschoren, Automated Machine Learning: Methods, Systems, Challenges (Springer, 2018).
40. R. D. King et al., The automation of science. Science 324, 85–89 (2009).
41. Wilkinson et al. The FAIR guiding principles for scientific data management and stewardship. Sci. Data 3, 160018 (2016).
42. A. Gangjee et al., Design, synthesis, and X-ray crystal structures of 2,4-diamino-furo[2,3-d]pyrimidines as multireceptor tyrosine kinase and dihydrofolate reductase inhibitors. Bioorg. Med. Chem. 17, 7324–7336 (2009).
43. R. F. Sugden, The action of indoramin and other compounds on the uptake of neurotransmitters into rat cortical slices. Br. J. Pharmacol. 51, 467–469 (1974).
44. European Medicines Agency, Assessment report, Alecensa, EMA/197343/2017 (2017). Accessed 1 August 2021.
45. K. Ghose, A. Coppen, D. Carroll, Intravenous tyramine response in migraine before and during treatment with indoramin. BMJ 1, 1191–1193 (1977).
46. R. Long, Y. Zhu, S. Zhou, Therapeutic role of melatonin in migraine prophylaxis: A systematic review. Medicine (Baltimore) 98, e14099 (2019).
47. N. R. Brinn, M. A. Gandhi, A review of the role of melatonin in irritable bowel syndrome. Am. J. Pharm. Health Res. 2, 1 (2014).