Imposing Hard Logical Constraints on Multi-label Classification Neural Networks

Abstract

Machine Learning, and in particular deep learning, is becoming increasingly ubiquitous, and it is likely to be applied in almost every aspect of our lives in the next few years. However, the careless application of such methods in the real world can have, and has already had, disastrous consequences. In order to avoid such undesirable and potentially dangerous scenarios, a standard approach is to formally specify the desired behavior of the system and then ensure its compliancy to the specified properties. In this paper, we thus propose to enhance deep learning models by incorporating background knowledge as hard logical constraints. The constraints rule out the models' undesired behaviors and can be exploited to gain better performance. In order to achieve the above, we propose CCN(h), a novel model for multi-label classification problems with hard constraints expressed as normal logic rules. Given any multi-label classification neural network h, CCN(h) is able to exploit the information expressed by the constraints to: (i) produce predictions that are guaranteed to satisfy the constraints, and (ii) improve the performances. We conduct an extensive experimental analysis showing the superior performance of CCN(h) when compared to state-of-the-art models in the setting of multi-label classification problems with hard logical constraints.

1 Introduction

Machine Learning, and in particular Deep Learning (DL), is becoming increasingly ubiquitous, and it is likely to be applied in almost every aspect of our lives in the next few years. This is happening thanks to the successes of DL, which allowed to solve problems that were previously thought to be unbreakable (see e.g., (Senior et al. 2020; Stokes et al. 2020)). Such results, though ground-breaking, overshadow the dangers that come with the careless application of DL models in the real world. Indeed, while these models report astonishingly high-performance in terms of accuracy (or alternatively chosen metric), they do not give any guarantee that the model will not have any unintended behaviour when used in practice.

In this paper, we thus propose to enhance DL models by incorporating background knowledge as hard logic constraints. The constraints rule out the models' undesired behaviors and can be exploited to gain better performance. In order to achieve the above, we propose *coherent-by-construction network*[1] CCN(h), a novel model for Multi-label Classification (MC) problems with hard constraints over the output space expressed as normal logic rules (Lloyd 1987), that is, expressions of the form:

$$A_1, \ldots, A_k, \neg A_{k+1}, \ldots, \neg A_n \rightarrow A, \quad (0 \leq k \leq n), \quad (1)$$

imposing that whenever the classes $A_1, \ldots, A_k$ are predicted, while $A_{k+1}, \ldots, A_n$ are not, then also the class A should be predicted. Such constraints generalize hierarchy constraints (corresponding to the case $n = k = 1$) used in the field of hierarchical MC problems. Indeed, consider, for example, the task proposed by (Dimitrovski et al. 2008), where a radiological image has to be annotated with an IRMA code specifying, among others, the biological system examined. In this setting, we will be able to specify not only that the prediction $\{\textit{stomach}\}$ alone is not possible thanks to the hierarchical constraint

$$\textit{stomach} \rightarrow \textit{gastrointestinalSystem}, \quad (2)$$

but we will be also able to capture constraints like "if a medical image contains the abdomen but neither the middle nor the upper abdomen, then it contains the lower abdomen", which can we written as:

$$\textit{abdomen}, \neg \textit{middleAbdomen}, \neg \textit{upperAbdomen} \rightarrow \textit{lowerAbdomen}.$$

CCN(h) is the first model able to deal with MC problems with such expressive hard constraints. CCN(h) can be built on top of any multi-label classification neural network h (to be trained) and is based upon two basic ingredients:

1. the *Constraint Module* (CM): a constraint layer built on top of h, which extends the predictions made by h in order to ensure that the predictions satisfy the constraints, and
2. the *Constraint Loss* (CLoss) a loss function, teaching CCN(h) when to exploit the constraints, that is, in the presence of (1), when to exploit the prediction on $\{A_1, \ldots, A_n\}$ to make predictions on A.

[1] As defined in (Giunchiglia and Lukasiewicz 2020), given a model m and a set of constraints Π, m is *coherent* with Π if all its output predictions satisfy Π.

We further show that, if we restrict ourselves to constraints with stratified negation (Apt, Blair, and Walker 1988), given a set $\mathcal{H}$ of initial predictions made by the underlying model h, CCN(h) is able to compute the unique minimal set of classes $\mathcal{M}$ such that

1. $\mathcal{M}$ extends $\mathcal{H}$, that is, such that $\mathcal{H} \subseteq \mathcal{M}$, and
2. $\mathcal{M}$ is coherent with (satisfies) the constraints, that is, such that, given (1), $A \in \mathcal{M}$ whenever $\{A_1, \ldots, A_k\} \subseteq \mathcal{M}$ and $\{A_{k+1}, \ldots, A_n\} \cap \mathcal{M} = \emptyset$.

Indeed, for a non-stratified set of constraints expressed as normal rules, there can be no or more than one minimal set of classes having the above two properties, and determining the non-existence or computing one of them can take exponential time.

2 Coherent-by-Construction Networks

Preliminaries and Notation Let Π be a set of hard constraints expressed as normal logic rules, h be a generic neural network for a MC problem, and $\mathcal{H}$ the set of classes predicted by h.

Given a constraint $r \in \Pi$, we call $head(r) = A$ the *head* of r, and $body(r) = body^+(r) \cup body^-(r)$ the *body* of r, where $body^+(r) = \{A_1, \ldots, A_k\}$ and $body^-(r) = \{\neg A_{k+1}, \ldots, \neg A_n\}$.

Definition 2.1 A set of constraints Π is *stratified* if there is a partition $\Pi_1, \Pi_2, \ldots, \Pi_s$ of Π, with Π_1 possibly empty, such that, for every $i \in \{1, \ldots, s\}$,

1. for every class $A \in \cup_{r \in \Pi_i} body^+(r)$, all the constraints with head A in Π belong to $\cup_{j=1}^{i} \Pi_j$;
2. for every $\neg A \in \cup_{r \in \Pi_i} body^-(r)$, all the constraints with head A in Π belong to $\cup_{j=1}^{i-1} \Pi_j$.

$\Pi_1, \Pi_2, \ldots, \Pi_s$ is a *stratification* of Π, and each Π_i is a *stratum*.

Given a rule like the one in Equation (1) and a generic model m we need to compute the values associated to $\neg A_{k+1}, \ldots, \neg A_n$ given the output of m. We call such values $\overline{m}_{A_{k+1}}, \ldots, \overline{m}_{A_n}$, and they must be such that for each class $A_i \in \{A_{k+1}, \ldots, A_n\}$ and threshold θ,

1. $\overline{m}_{A_i} = 1$ when $m_{A_i} = 0$, and $\overline{m}_{A_i} = 0$ when $m_{A_i} = 1$,
2. $\overline{m}_{A_i}$ is strictly decreasing and continuous (small changes to the value of m_{A_i} should correspond to small changes in the value of $\overline{m}_{A_i}$), and
3. $\overline{m}_{A_i} = \theta$ when $m_{A_i} = \theta$.

For any threshold θ there are infinitely many functions $\overline{v}$ satisfying such requirements. A simple solution is to require $\overline{v}$ to be piecewise linear with two segments joining when $\overline{v} = v = \theta$, in which case, if $\theta = 0.5$, we obtain $\overline{v} = 1 - v$, that is the *standard negation* in fuzzy logics (Metcalfe 2005). For simplicity, from here on, we assume to have the standard negation, that is, to fix the threshold θ to 0.5 and $\overline{v} = 1 - v$, for each $v \in [0, 1]$. All the definitions and results generalize to the case in which we have an arbitrary strict negation with $\overline{\theta} = \theta$.

Constraint Module Given h and a set of stratified constraints Π, our goal is to get CCN(h) to predict a final set of classes $\mathcal{M}$ such that:

1. $\mathcal{M}$ extends the set of classes $\mathcal{H}$ predicted by h (i.e., $\mathcal{H} \subseteq \mathcal{M}$) and be coherent with Π;
2. $\mathcal{M}$ is such that any class in $\mathcal{M} \setminus \mathcal{H}$ is in the head of a constraint r in Π, with the chain of rules used to satisfy $body(r)$ grounded in $\mathcal{H}$;
3. $\mathcal{M}$ includes only those classes that are either in $\mathcal{H}$ or are forced to be in $\mathcal{M}$ through the explicit use of chains of rules grounded in $\mathcal{H}$;
4. $\mathcal{M}$ is unique, that is, there will be no other set of classes $\mathcal{M}'$ satisfying the above requirements.

The first property is the obvious one: the constraints in Π must be satisfied, and CCN(h) can only derive more classes in the head of the constraints. Whereas the second property is formalized by the concept of supportedness as defined in the work by (Apt, Blair, and Walker 1988).

Definition 2.2 A set of classes $\mathcal{M}$ is *supported* relative to $\mathcal{H}$ and Π, if for any class $A \in \mathcal{M}$, $A \in \mathcal{H}$, or there exists a constraint $r \in \Pi$ such that $head(r) = A$, $body^+(r) \subseteq \mathcal{M}$, and for each $\neg B \in body^-(r)$, we have $B \notin \mathcal{M}$.

The third property is a minimality condition.

Definition 2.3 $\mathcal{M}$ is *minimal* relative to $\mathcal{H}$ and Π, if there exists no set of classes $\mathcal{M}'$ with $\mathcal{H} \subseteq \mathcal{M}' \subset \mathcal{M}$ that is coherent with Π.

The four properties together ensure that the final predictions made by CCN(h) are coherent with Π and that can be uniquely explained on the grounds of the initial predictions made by h and the constraints in Π.

Assuming Π is stratified, in order to compute the output of CCN(h), we need to first compute the stratification of Π in the following way:

1. we compute the acyclic component graph (Cormen et al. 2009) of the dependency graph G_Π of Π (as defined in (Apt, Blair, and Walker 1988)), that is, the DAG obtained by shrinking each strongly connected component in G_Π into a single vertex (notice that since Π is stratified, negative edges are not involved in any cycle in G_Π),
2. we assign to the classes in each node of the DAG the number 1 plus the maximum number of negative edges connecting a root to the node, and
3. we define:
   (a) $\mathcal{A}_i$ as the set of classes having the number i assigned at the previous step, and
   (b) Π_i as the set of constraints in Π whose head is in $\mathcal{A}_i$.

   Define Π_i^* to be the set of constraints

   1. initially equal to Π_i, and then
   2. obtained by recursively adding the constraints obtained from a constraint r already in Π_i^* by substituting a class $A \in body^+(r) \cap \mathcal{A}_i$ with $body(r')$ for any rule r' with $head(r') = A$ (hence, $r' \in \Pi_i$), and finally

3. eliminating the constraints r such that $head(r) \in body^+(r)$, or for which there exists another constraint $r' \in \Pi_i$ with $head(r) = head(r')$ and $body(r') \subset body(r)$.

Π_i^* is guaranteed to be finite, since the set of classes $\mathcal{A}$ is finite, and we do not allow for repetitions in the body of constraints. The constraints being eliminated in the third step are redundant.

Then, for each class $A \in \mathcal{A}_i$, we define the output CM_A of the constraint module CM via

$$\text{CM}_A = \max(h_A, h_A^{r_1}, \ldots, h_A^{r_p}), \tag{3}$$

where

1. $r_1, \ldots, r_p$ are all the constraints in Π_i^* with head A, and
2. assuming $r \in \Pi_i^*$ has the form (1),

$$h_A^r = \min(v_{A_1}, \ldots, v_{A_k}, \overline{\text{CM}}_{A_{k+1}}, \ldots, \overline{\text{CM}}_{A_n}),$$

with $v_{A_1} = h_{A_1}$ if $A_1 \in \mathcal{A}_i$, and $v_{A_1} = \text{CM}_{A_1}$ if $A_1 \in \cup_{j=1}^{i-1}\mathcal{A}_j$. Analogously for $v_{A_2}, \ldots, v_{A_k}$.

The above definition is well-founded:

1. Π_1^* does not contain negated classes, and thus the definition of CM_A when $A \in \mathcal{A}_1$ relies only on the outputs of the bottom module h, and
2. the definition of CM_A when $A \in \mathcal{A}_i$ $(i > 1)$ uses only outputs of h or of already defined outputs of CM.

Given the above the definition, we can now state that CCN(h) has the desired properties.

Theorem 2.4 *Assume Π is stratified, and let $\mathcal{M}$ be the set of classes predicted by* CCN(h)*. Then, $\mathcal{M}$*

1. *extends $\mathcal{H}$ and is coherent with Π,*
2. *is supported relative to $\mathcal{H}$ and Π,*
3. *is minimal relative to $\mathcal{H}$ and Π, and*
4. *is the unique set satisfying the previous properties.*

Constraint Loss For every data point, the value of the loss function CLoss used to train CCN(h) is defined as:

$$\text{CLoss} = \sum_{A \in \mathcal{A}} \text{CLoss}_A,$$

CLoss_A being the value of the loss for class A, defined as:

$$\text{CLoss}_A = -y_A \ln(\text{CM}_A^+) - \overline{y}_A \ln(\overline{\text{CM}}_A^-),$$

where:

- y_A is the ground truth for class A,
- CM_A^+ is the value to optimize when $y_A = 1$, and
- CM_A^- is the value to optimize when $y_A = 0$.

CM_A^+ and CM_A^- differ from the output value CM_A of CCN(h) for class A, that is, from CCN(h)$_A$.

Consider a class A in the ith stratum (i.e., $A \in \mathcal{A}_i$). To each constraint r with head A in Π_i^* we associate two values

1. $h_A^{+,r}$ to be used with $y_A = 1$, and

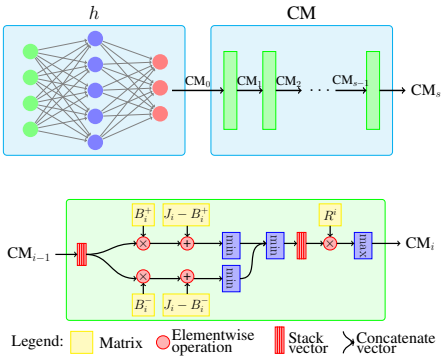

Figure 1: Visual representation of CCN(h) (left), and details of the operations associated with each stratum (right).

2. $h_A^{-,r}$ to be used with $y_A = 0$.

Assume $y_A = 1$. Consider a constraint r in Π_i^* with head A of the form (1). Then, we want to teach CCN(h) to possibly exploit the constraint r for predicting A if $y_{A_1} = \ldots = y_{A_k} = 1$ and $y_{A_{k+1}} = \ldots = y_{A_n} = 0$. We thus define,

$$h_A^{+,r} = \min(v_{A_1} y_{A_1}, \ldots, v_{A_k} y_{A_k}, \overline{v}_{A_{k+1}} \overline{y}_{A_{k+1}}, \ldots, \overline{v}_{A_n} \overline{y}_{A_n}),$$

where v_{A_l} is

1. h_{A_l} if $A_l \in \mathcal{A}_i$ (and thus $1 \leq l \leq k$),
2. $\text{CM}_{A_l}^+$ if $A_l \in \cup_{j=1}^{i-1}\mathcal{A}_j$ and $1 \leq l \leq k$,
3. $\text{CM}_{A_l}^-$ if $A_l \in \cup_{j=1}^{i-1}\mathcal{A}_j$ and $k+1 \leq l \leq n$.

The value CM_A^+ associated to class A when $y_A = 1$ is

$$\text{CM}_A^+ = \max(h_A, h_A^{+,r_1}, \ldots, h_A^{+,r_p}),$$

where $r_1, \ldots, r_p$ are all the constraints in Π_i^* with head A.

Assume $y_A = 0$. Consider a constraint r in Π_i^* with head A of the form (1). Then, for some class $A_l \in body^+(r)$ $y_{A_l} = 0$, or for some class $A_l \in body^-(r)$ $y_{A_l} = 1$, and we want to teach CCN(h) to not fire the constraint r. We thus define

$$h_A^{-,r} = \min(v_{A_1} \overline{y}_{A_1} + y_{A_1}, \ldots, v_{A_k} \overline{y}_{A_k} + y_{A_k}, \overline{v}_{A_{k+1}} y_{A_{k+1}} + \overline{y}_{A_{k+1}}, \ldots, \overline{v}_{A_n} y_{A_n} + \overline{y}_{A_n}),$$

where v_{A_l} now is

1. h_{A_l} if $A_l \in \mathcal{A}_i$ (and thus $1 \leq l \leq k$),
2. $\text{CM}_{A_l}^-$ if $A_l \in \cup_{j=1}^{i-1}\mathcal{A}_j$ and $1 \leq l \leq k$,
3. $\text{CM}_{A_l}^+$ if $A_l \in \cup_{j=1}^{i-1}\mathcal{A}_j$ and $k+1 \leq l \leq n$.

The value CM_A^- associated with the class A when $y_A = 0$ is

$$\text{CM}_A^- = \max(h_A, h_A^{-,r_1}, \ldots, h_A^{-,r_p}),$$

where $r_1, \ldots, r_p$ are all the constraints in Π_i^* with head A.

3 GPU Implementation

Constraint Module The basic idea, starting from the vector CM_0 (which contains the l values resulting from the bottom module h) is to iteratively compute the vector of values

CM_i corresponding to the outputs of CM if given the set of constraints $\cup_{j=1}^{i}\Pi_j^*$. The final output of CM will correspond to CM_s. Figure 1 shows a visual representation of the process and of CCN(h).

For each $i \in \{1, \ldots, s\}$, p_i is the number of constraints in Π_i^* ($p_i = |\Pi_i^*|$), r_{ij} denotes the jth constraint in Π_i^*, and

- B_i^+ is the $p_i \times l$ matrix whose j, k element is 1 if $A_k \in body^+(r_{ij})$, and 0 otherwise;
- B_i^- is the $p_i \times l$ matrix whose j, k element is 1 if $\neg A_k \in body^-(r_{ij})$, and 0 otherwise;
- C_{i-1} is the $p_i \times l$ matrix obtained by stacking p_i times CM_{i-1}.

Stacking p times a vector v of size q returns the $p \times q$ matrix $1_p^T \times v$ whose j, k element is $v[k]$.

Then, the jth value $v_i[j]$ of the vector v_i associated with the body of the constraint r_{ij} is

$$v_i^+ = \min(B_i^+ \odot C_{i-1} + (J_{p_i,l} - B_i^+), \text{dim} = 1),$$
$$v_i^- = \min(B_i^- \odot (J_{p_i,l} - C_{i-1}) + (J_{p_i,l} - B_i^-), \text{dim} = 1),$$
$$v_i[j] = \min(v_i^+[j], v_i^-[j]),$$

where

1. $\odot$ represents the Hadamard product,
2. $J_{p,q}$ is the $p \times q$ matrix of ones, and
3. given an arbitrary $p \times q$ matrix Q, $\min(Q, \text{dim} = 1)$ (resp., $\max(Q, \text{dim} = 1)$) returns a vector of length p whose ith element is equal to $\min(Q_{i1}, \ldots, Q_{iq})$ (resp., $\max(Q_{i1}, \ldots, Q_{iq})$).

Then, the output of the ith layer associated with the ith stratum is given by:

$$\text{CM}_i = \max(IH_i \odot V_i, \text{dim} = 1),$$

where

- H_i is the $l \times p_i$ matrix whose j, k element is 1 if $A_j = head(r_{ik})$, and 0 otherwise,
- IH_i is the $l \times (l + p_i)$ matrix obtained by stacking the $l \times l$ identity matrix and H_i,
- V_i is the $l \times (l + p_i)$ matrix obtained by stacking l times the concatenation of CM_{i-1} and v_i.

Constraint Loss The computation of constraint loss on GPU follows the same reasoning of the constraint module.

4 Experimental Analysis

In order to prove the superiority of our model, we adopted the same methodology presented in (Feng, An, and He 2019) and consider the three well-established MC models and the state-of-the-art MC model tested in (Feng, An, and He 2019), which can be characterized by the order of classes correlations they exploit. Thus, CCN(h) is compared with

1. BR (Boutell et al. 2004), a first order model which considers each class separately, ignoring class correlations, and
2. ECC (Read et al. 2009), RAKEL (Tsoumakas et al. 2009) and CAMEL (Feng, An, and He 2019), which exploit correlations among two or more classes.

Metric	**Average ranking**					**Friedman test**
	CCN(h)	CAMEL	ECC	BR	RAKEL	
Average Prec.	**1.50**	2.17	3.53	3.58	4.22	✓✓
Coverage Err.	**1.22**	2.36	3.61	3.92	3.89	✓✓
Hamming Loss	**1.36**	2.56	3.39	3.61	3.92	✓✓
Multi-label Acc.	**1.69**	3.58	3.19	3.27	3.25	✓✓
One-error	**1.50**	2.08	3.67	3.56	4.19	✓✓
Ranking Loss	**1.22**	2.19	3.61	3.72	4.17	✓✓

Table 1: Average ranking for each metric and model, results of the Friedman. We use ✓✓ to indicate that the test returned p-value < 0.01.

BR, ECC, and RAKEL are the well-established MC models, and CAMEL is the current state-of-the-art MC model (Feng, An, and He 2019). Since these models are not guaranteed to output predictions that are coherent with the constraints, we applied CM as additional post-processing steps. We tested the models on 16 publicly available datasets.

About the metrics, the analysis of 64 papers on MC problems conducted by (Spolaôr et al. 2013) and reported by (Pereira et al. 2018), shows that already in 2013 as many as 19 different metrics have been used to evaluate MC models, and still today different papers use different subsets of such metrics. However, as suggested by (Pereira et al. 2018), not all subsets can be used, as the experimental results may appear to favor a specific behavior depending on the subset of measures chosen, thus possibly leading to misleading conclusions. To avoid such undesired results, the authors conducted a correlation analysis of the metrics that led to the individuation of clusters of correlated measures and thus to the proposal of various subsets of metrics. Following the above criteria, we used the six metrics listed in Table 1. Due to lack of space, we cannot report the results of each model, however, in Table 1, the average rankings of each model can be found,[2] and we can see that CCN(h) outranks all the other models on all metrics. We also verified the statistical significance of the results—as advised by (Demsar 2006)—by performing the Friedman test for each metric. As reported in the last column of Table 1, the Friedman test returned p-value < 0.01 for each metric.

5 Conclusions

In this paper, we proposed to enhance DL models by incorporating background knowledge as hard logic constraints. To this end, we developed CCN(h), a novel model whose predictions are always coherent with the given constraints and such that it is able to exploit the background knowledge expressed by the constraints to gain better performance, as demonstrated by the fact that it outperforms the state-of-the-art MC models on 16 datasets. The paper is an extended abstract of (Giunchiglia and Lukasiewicz 2021).

[2]Since we are considering rankings, for each metric we have that the lower the better.

References

Apt, K. R.; Blair, H. A.; and Walker, A. 1988. Towards a theory of declarative knowledge. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann. 89–148.

Boutell, M. R.; Luo, J.; Shen, X.; and Brown, C. M. 2004. Learning multi-label scene classification. *Pattern Recognition* 37(9).

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2009. *Introduction to Algorithms, 3rd Edition*. MIT Press.

Demsar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7:1–30.

Dimitrovski, I.; Kocev, D.; Loskovska, S.; and Džeroski, S. 2008. Hierchical annotation of medical images. In *Proceedings of IS*, 174–181.

Feng, L.; An, B.; and He, S. 2019. Collaboration based multi-label learning. In *Proceedings of AAAI*, 3550–3557.

Giunchiglia, E., and Lukasiewicz, T. 2020. Coherent hierarchical multi-label classification networks. In *Proceedings of NeurIPS*.

Giunchiglia, E., and Lukasiewicz, T. 2021. Multi-label classification neural networks with hard logical constraints. *J. Artif. Intell. Res.* 72:759–818.

Lloyd, J. W. 1987. *Foundations of Logic Programming, 2nd Edition*. Springer.

Metcalfe, G. 2005. Fundamentals of fuzzy logics. https://www.logic.at/tbilisi05/Metcalfe-notes.pdf.

Pereira, R. B.; Plastino, A.; Zadrozny, B.; and Merschmann, L. H. 2018. Correlation analysis of performance measures for multi-label classification. *Information Processing & Management* 54(3):359 – 369.

Read, J.; Pfahringer, B.; Holmes, G.; and Frank, E. 2009. Classifier chains for multi-label classification. In Buntine, W.; Grobelnik, M.; Mladenić, D.; and Shawe-Taylor, J., eds., *Machine Learning and Knowledge Discovery in Databases*, 254–269. Berlin, Heidelberg: Springer Berlin Heidelberg.

Senior, A. W.; Evans, R.; Jumper, J.; Kirkpatrick, J.; Sifre, L.; Green, T.; Qin, C.; Žídek, A.; Nelson, A. W. R.; Bridgland, A.; Penedones, H.; Petersen, S.; Simonyan, K.; Crossan, S.; Kohli, P.; Jones, D. T.; Silver, D.; Kavukcuoglu, K.; and Hassabis, D. 2020. Improved protein structure prediction using potentials from deep learning. *Nature* 577(7792):706–710.

Spolaôr, N.; Cherman, E. A.; Metz, J.; and Monard, M. C. 2013. A systematic review on experimental multi-label learning. Technical Report 362, Institute of Mathematics and Computational Sciences, University of São Paulo.

Stokes, J. M.; Yang, K.; Swanson, K.; Jin, W.; Cubillos-Ruiz, A.; Donghia, N. M.; MacNair, C. R.; French, S.; Carfrae, L. A.; Bloom-Ackermann, Z.; Tran, V. M.; Chiappino-Pepe, A.; Badran, A. H.; Andrews, I. W.; Chory, E. J.; Church, G. M.; Brown, E. D.; Jaakkola, T. S.; Barzilay, R.; and Collins, J. J. 2020. A deep learning approach to antibiotic discovery. *Cell* 180(4):688–702.e13.

Tsoumakas, G.; Dimou, A.; Spyromitros, E.; Mezaris, V.; Kompatsiaris, I.; and Vlahavas, I. 2009. Correlation-based pruning of stacked binary relevance models for multi-label learning. In *Proceedings of International Workshop on Learning from Multi-label Data*.