Taming the Sigmoid Bottleneck: Provably Argmaxable Sparse Multi-Label Classification

Andreas Grivas, Antonio Vergari†, Adam Lopez†

School of Informatics, University of Edinburgh, UK
{agrivas, avergari, alopez}@ed.ac.uk

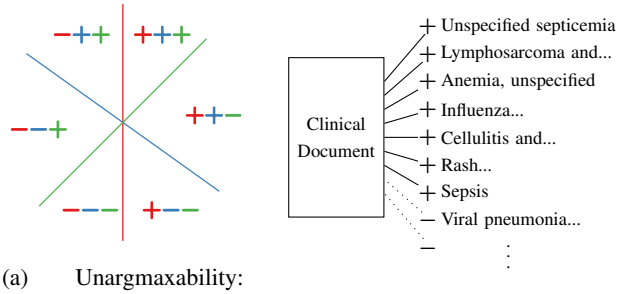
Abstract

Sigmoid output layers are widely used in multi-label classification (MLC) tasks, in which multiple labels can be assigned to any input. In many practical MLC tasks, the number of possible labels is in the thousands, often exceeding the number of input features and resulting in a *low-rank* output layer. In multi-class classification, it is known that such a low-rank output layer is a bottleneck that can result in *unargmaxable* classes: classes which cannot be predicted for any input. In this paper, we show that for MLC tasks, the analogous *sigmoid bottleneck* results in exponentially many unargmaxable label combinations. We explain how to detect these unargmaxable outputs and demonstrate their presence in three widely used MLC datasets. We then show that they can be prevented in practice by introducing a Discrete Fourier Transform (DFT) output layer, which guarantees that all sparse label combinations with up to k active labels are argmaxable. Our DFT layer trains faster and is more parameter efficient, matching the F1@k score of a sigmoid layer while using up to 50% fewer trainable parameters. Our code is publicly available at https://github.com/andreasgrv/sigmoid-bottleneck.

1 Introduction

Sigmoid classifiers for Multi-Label Classification (MLC) are simple to implement: just append a linear layer with sigmoid activations to your neural feature encoder of choice. They are widely used in neural MLC with thousands of output labels; applications include clinical coding (Mullenbach et al. 2018), image classification (Baruch et al. 2020), fine-grained entity typing (Choi et al. 2018) and protein function prediction (Kulmanov and Hoehndorf 2019). Moreover, they are the default for MLC in frameworks such as Scikit-learn (Pedregosa et al. 2011) and Keras (Paul and Rakshit 2014). In this paper we highlight an overlooked weakness of this layer. If, as is common for computational efficiency, we make the number of features smaller than the number of labels, the result is a **Bottlenecked Sigmoid Layer (BSL)**, in which *exponentially many label combinations cannot be predicted irrespective of input*. We say that such outputs are **unargmaxable** (Grivas, Bogoychev, and Lopez 2022). Fig. 1a illustrates how a BSL with two features and three labels must have unargmaxable label combinations.

† Co-supervision.



(a) Unargmaxability: $+-+$ and $-+-$ cannot be predicted.

(b) Unargmaxable test example from MIMIC-III for BSL with $d = 50$.

Figure 1: When we have more labels (n) than features (d), some label combinations are unargmaxable, i.e. impossible to predict. Left: in a $d = 2$ feature space with $n = 3$ classification hyperplanes through the origin, only 6 out of 8 label combinations can be predicted irrespective of how we orient the hyperplanes.[1] Right: a BSL trained on the MIMIC-III clinical MLC dataset with $d = 50$ and $n = 8921$ is unable to predict this label combination[2] which has the depicted 7 active labels $(+)$ and the remaining ones are inactive $(-)$.

But unargmaxable label combinations are only a problem if our application requires those combinations. As we show in this paper, they often do. For example, in the safety-critical application of clinical MLC (see Section 5), unargmaxability can make it impossible to label a report with a specific combination of findings, as illustrated with a real example in Fig. 1b. This would be surprising and unacceptable to users of the system when such label combinations do indeed occur in data. Since BSLs are widely used, it is critical for developers and users of a model to be aware of this problem and to be able to guarantee that all meaningful outputs for the task at hand are argmaxable.

Previous work has shown that bottlenecked output layers have restrictions on expressivity, but focused only on multi-

[1] Adding a bias term to the BSL allows the hyperplanes to have offsets and they will not necessarily meet at the origin. However, this cannot solve the problem: such a BSL is more restricted than increasing d by one: we still only get 7 out of 8 label combinations.

[2] There are also unargmaxable test examples for $d = 100$ and $d = 200$, we chose this example as it had fewer active labels.

class classification (Yang et al. 2018; Ganea et al. 2019) and not MLC. But for multi-class classification, the consequences are minor: classes can be unargmaxable in theory (Demeter, Kimmel, and Downey 2020) but rarely are in practice (Grivas, Bogoychev, and Lopez 2022). In MLC, we will show that exponentially many label combinations are unargmaxable, and as we have already seen in Fig. 1b, meaningful outputs can be unargmaxable. While a BSL can in principle learn to represent any particular output, it comes with no guarantees. And although we can obtain post hoc guarantees by verifying whether specific meaningful outputs are argmaxable, there may be too many outputs to check exhaustively. To sidestep this limitation, we show how to construct an output layer that guarantees that meaningful outputs are argmaxable by construction. To do so, we provide guarantees for a superset of outputs: those with up to k active labels, where we choose k based on the statistics of the dataset. This is possible since for most MLC tasks k is bounded (Jain et al. 2019), either empirically (e.g. k=80 for MIMIC-III) or by construction (e.g. k=50 for BioASQ (Tsatsaronis et al. 2015)).

In summary, our contributions are: **i)** We formalise the argmaxability problem for MLC and expose the limitations of BSLs which are widely used in practice (Section 2); **ii)** We provide ways of verifying if a label combination is argmaxable for a model (Section 3) and show that for three widely used MLC datasets BSLs can have unargmaxable test set label combinations (Section 5). **iii)** We prove that this need not be the case; we can guarantee that any output with up to k active labels is argmaxable by constraining the output layer parametrisation to a family of matrices. The Discrete Fourier Transform (DFT) matrix is in this family and we use it to parametrise our DFT layer, an efficient replacement output layer with such guarantees (Section 4). **iv)** Through experiments on three MLC datasets we show that our DFT layer guarantees that meaningful outputs are argmaxable while converging faster and being more parameter efficient than a BSL (Section 5).

2 Multi-label Classification

We consider a MLC model that predicts a complete label assignment $\mathbf{y} \in \{+,-\}^n$ for a label vocabulary of size n, and where each $\mathbf{y}_i \in \{+,-\}$ denotes if a single label is active (+) or inactive (-). Many neural MLC models for problems with large label vocabularies employ an output layer that is linear, e.g., for fine-grained entity typing (Choi et al. 2018), protein function prediction (Kulmanov and Hoehndorf 2019), clinical coding (Mullenbach et al. 2018) and multi-label image classification (Baruch et al. 2020).

Bottlenecked Sigmoid Layers. A linear sigmoid layer takes as input a feature vector $\mathbf{x} \in \mathbb{R}^d$ and predicts $\mathbf{y}$ by assuming that all labels are independent given $\mathbf{x}$. The idea is that a powerful encoder does the "heavy lifting" and projects inputs to meaningul embeddings in $\mathbb{R}^d$ such that they can be easily separated by n different hyperplanes. When n is large, due to computational constraints it is popular to realise such a layer as a *Bottlenecked Sigmoid Layer* (BSL) which is parametrised by a low-rank $\mathbf{W} \in \mathbb{R}^{n \times d}$ and associates with each label the probability $P(\mathbf{y}_i \mid \mathbf{x}) = \sigma\left(\mathbf{w}^{(i)\top}\mathbf{x}\right)$ if $\mathbf{y}_i = +$ and $1 - \sigma\left(\mathbf{w}^{(i)\top}\mathbf{x}\right)$ otherwise. Here, σ is the logistic sigmoid and $\mathbf{w}^{(i)}$ is the weight vector of the i-th classifier (hyperplane), i.e., the i^{th} row of $\mathbf{W}$. Note that all $\mathbf{w}^{(i)}$ see the same shared $\mathbf{x}$. We focus on such a setup and discuss its limitations because it is the default in mainstream ML libraries such as scikit-learn (Pedregosa et al. 2011) and is largely used as a simple classifier (Mullenbach et al. 2018; Baruch et al. 2020; Kulmanov and Hoehndorf 2019). In the following, we will denote a whole multi-label classifier by the parametrization of its last layer, e.g., we will say "a classifier $\mathbf{W}$", as our analysis is agnostic to the feature encoder.

Argmaxable Label Assignments. Making a prediction with a BSL boils down to predicting every label independently by computing $\mathbf{y}_i^* = \operatorname{argmax}_{\mathbf{y}_i} P(\mathbf{y}_i \mid \mathbf{x})$. This is equivalent to taking the sign of the logit of the i-th classifier, i.e., computing $\mathbf{y}_i^* = \operatorname{sign}\left(\mathbf{w}^{(i)\top}\mathbf{x}\right)$ where $\operatorname{sign}(\mathbf{z}_i) = +$ if $\mathbf{z}_i > 0$ and $-$ if $\mathbf{z}_i < 0$. [3] Therefore, $\mathbf{y}^* = \operatorname{sign}(\mathbf{W}\mathbf{x})$.

Definition 1. *A label assignment* $\mathbf{y}$ *is* ***argmaxable*** *for a classifier* $\mathbf{W}$ *if there exists an input* $\mathbf{x}$ *for which thresholding the output probabilities using the argmax decision rule* $\operatorname{sign}\left(P(\mathbf{y}_i = + \mid \mathbf{x}) - \frac{1}{2}\right)$ *produces* $\mathbf{y}_i$,[4] *i.e.* $\mathbf{y}$ *is argmaxable* $\iff \exists \mathbf{x} : \operatorname{sign}(\mathbf{W}\mathbf{x}) = \mathbf{y}$.

From a geometric perspective, we can interpret the n rows of $\mathbf{W}$ as the normal vectors, $\mathbf{w}^{(i)}$, of n hyperplanes that tesselate feature space into regions. We can identify each region by assigning it a sign vector which identifies on which side of each hyperplane it is (Fig. 1, left). From this view, a label assignment $\mathbf{y}$ is argmaxable if the halfspaces intersect in such a way that the corresponding region is formed. For example, the region – – – is formed as an intersection of the negative halfspaces. For a classifier parameterised by a full rank $\mathbf{W} \in \mathbb{R}^{n \times n}$, all 2^n label combinations are argmaxable. However, since $d \ll n$ in many real-world applications, as discussed in Section 1, $\mathbf{W}$ will be low rank, and therefore *only a (small) subset of label combinations is argmaxable*.

2.1 Hyperplane Overcrowding

For a low-rank classifier $\mathbf{W} \in \mathbb{R}^{n \times d}$ and $d \ll n$, we have a large number of hyperplanes finely slicing a lower dimensional feature space. The natural question is, *out of the* 2^n *label combinations* $\mathbf{y}$*, how many can such a classifier actually represent?* To elaborate on this, let us define the set of argmaxable label combinations for a classifier $\mathbf{W}$ as:

$$\mathcal{A}(\mathbf{W}) = \{\operatorname{sign}(\mathbf{W}\mathbf{x}) \quad \forall \mathbf{x} \in \mathbb{R}^d\} \tag{1}$$

We can exactly count the number of argmaxable label combinations, i.e. $|\mathcal{A}(\mathbf{W})|$ if $\mathbf{W}$ is in **general position**.

Definition 2. $\mathbf{W} \in \mathbb{R}^{n \times d}$ *is in* ***general position*** *if no subset of* d *rows is linearly dependent. See Appendix A.*

Theorem 1. *(Cover 1965, Thm 2) If* $\mathbf{W}$ *is in general position, the number of argmaxable label combinations is:*

$$|\mathcal{A}(\mathbf{W})| = 2\sum_{d'=0}^{d-1} \binom{n-1}{d'}. \tag{2}$$

[3] In the case $\mathbf{z}_i = 0$, we perturb $\mathbf{x}$ by a tiny $\boldsymbol{\epsilon}$ and recompute $\mathbf{z}$.

[4] Due to monotonicity of sigmoid $\sigma(a) > \frac{1}{2} \iff a > 0$.

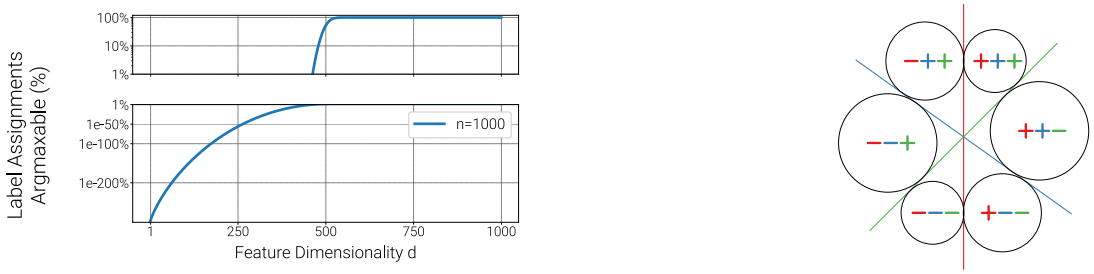


Figure 2: Left: We log-plot what percentage of the 2^{1000} label combinations is argmaxable for a BSL with $n = 1000$ labels as we decrease the feature dimensionality d (from right to left). When $d \ll n$ we can represent exponentially fewer outputs. We split the y-axis to highlight the fast dip when $d < 500$. Right: Our $n = 3, d = 2$ example from Fig. 1. We include the balls found by the Chebyshev LP for each argmaxable label combination. When $d \ll n$, most balls will have a tiny radius.

It follows that i) the number of argmaxable label combinations depends only on n and d, not the specific $\mathbf{W}$, and ii) most label combinations will be unargmaxable for $d \ll n$ as Eq. (2) indicates an exponential growth (see Fig. 2, Left).

While we can count the number of (un)argmaxable label combinations, it remains an open problem to verify if a specific set of label combinations can ever be predicted by a given classifier. We provide a solution in the next section.

3 Verifying Argmaxable Label Assignments

Given a low-rank classifier $\mathbf{W}$, we are interested in verifying if a set of L label combinations of interest $\{\mathbf{y}^{(l)}\}_{l=1}^{L}$ is (un)argmaxable. These labels can belong to a held out set, as in our experiments (Section 5), and help quantify the generalisability and trustworthiness of the given classifier, as we would expect it to be able to predict all L outputs.

A simple strategy is to verify the argmaxability of each $\mathbf{y}^{(l)}$. For that, we use a Chebyshev Linear Programme (LP), which also gives us a proxy for the size of a region, as we explain next. The LP aims to find the Chebyshev center (Boyd and Vandenberghe 2004, p. 417) of the region encoded by $\mathbf{y}^{(l)}$: the center of the largest ball of radius ϵ that can be embedded within it (see Fig. 2, Right). As a constrained optimization problem (see derivation in Appendix K)[5], we solve:

$$\begin{aligned} &\text{maximise} \quad \epsilon \qquad\qquad\qquad\qquad\qquad\qquad (3)\\ &\text{subject to} \quad -\mathbf{y}_i \mathbf{w}^{(i)\top}\mathbf{x} + \epsilon\|\mathbf{w}^{(i)}\|_2 \leq 0, \quad 1 \leq i \leq n,\\ &\qquad\qquad -10^4 \leq \mathbf{x}_j \leq 10^4, \quad 1 \leq j \leq d, \quad \epsilon > \text{eps} \end{aligned}$$

where we abuse notation and $\mathbf{y}_i \in \{+1, -1\}$. We constrain each entry of $\mathbf{x}$ in a bounded region, since the Chebyshev center is not defined otherwise. If the LP is feasible it returns the maximum radius ϵ and we verify that $\mathbf{y}$ is argmaxable. Note that we add an additional constraint, $\epsilon >$ eps, where eps is within the numerical accuracy the LP solver can operate.[6] As such, while we defined argmaxability in absolute terms (Section 2), in practice it can only be verified up to some numerical precision: $\mathbf{y}$ could be argmaxable, but an LP might not be able to detect it if the neighbourhood around all representative $\mathbf{x}$ is tiny. As such, we define ϵ-argmaxability to characterise robustly argmaxable label combinations.

Definition 3. *A label assignment* $\mathbf{y}$ *is* ***ϵ-argmaxable*** *for a classifier* $\mathbf{W}$ *if it is argmaxable even under the presence of any noise vector* $\boldsymbol{\delta}$ *having magnitude* $\|\boldsymbol{\delta}\|_2 \leq \epsilon$*:*

$$\mathbf{y} \textit{ is } \epsilon\textit{-argmaxable} \iff \forall \boldsymbol{\delta}, \|\boldsymbol{\delta}\|_2 \leq \epsilon, \exists \mathbf{x} : \operatorname{sign}(\mathbf{W}(\mathbf{x} + \boldsymbol{\delta})) = \mathbf{y}.$$

Our Chebyshev LP is able to verify ϵ-argmaxability, and therefore argmaxability, as the first implies the second. Note, however, that the reverse is not true. Although verifying that a classifier is able to argmax a certain set of labels is of extreme importance, verification can be computationally expensive, as LPs become intractable as we scale n, d and L. To avoid this, we devise a classifier that guarantees that all labels of interest are argmaxable *by design*.

4 DFT Layers for k-Active MLC

Designing a low-rank BSL that guarantees argmaxability for all 2^n possible label combinations is impossible, according to Theorem 1. However, for most MLC datasets the label combinations are *sparse*; only a handful of labels are *active* for any given example. As such, herein we choose an upper bound k on the number of active labels for each dataset and show how to modify the parametrisation of a BSL so that any k-active label assignment is guaranteed to be argmaxable. We first define sufficient criteria by specifying a *broad family of parametrisations* for which our result holds: the weight matrix should have at least $2k + 1$ input features and all its maximal minors should be non-zero and have the same sign. Next, we *specify* an implementation satisfying these criteria, the Discrete Fourier Transform (DFT) layer, which is computationally appealing and is accurate in practice.

4.1 k-Active Label Assignments

Label combinations in real-world MLC datasets are often sparse (Babbar and Schölkopf 2019); it is unlikely an image will contain more than k objects or that a clinical document will be assigned more than k clinical codes, where $k \approx \mathcal{O}(\log n)$ is a dataset dependent upper bound on the number of active labels (Jain et al. 2019). We now show how to guarantee that all k-active outputs are argmaxable by controlling the parametrisation of a BSL. We first formalise what a k-active label combination is below.

[5] See appendix at https://arxiv.org/abs/2310.10443.

[6] We use eps $= 10^{-8}$ since Gurobi Optimization (2021) has a minimum tolerance of 10^{-9}.

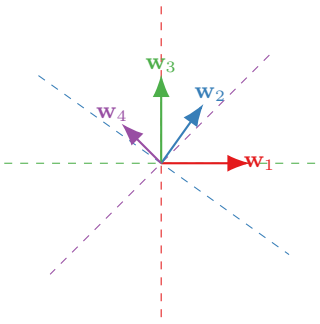
$$\mathbf{W} = \begin{bmatrix} 1.0 & 0.0 \\ 0.5 & 0.7 \\ 0.0 & 1.0 \\ -0.5 & 0.5 \end{bmatrix}$$

$\Delta_{\{1,2\}} = .7$ $\quad \Delta_{\{2,3\}} = .5$

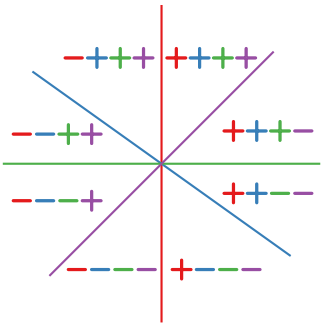
$\Delta_{\{1,3\}} = 1.$ $\quad \Delta_{\{2,4\}} = .6$

$\Delta_{\{1,4\}} = .5$ $\quad \Delta_{\{3,4\}} = .5$

(a) All $\binom{4}{2} = 6$ maximal minors, Δ_I, are positive, hence $\mathbf{W} \in \mathsf{Gr}^+_{4,2}$.



(b) Correspondence of $\mathbf{W}$ and geometric picture: each normal vector is a row of $\mathbf{W}$.



(c) The Argmaxable label assignments are the 1-alternating vectors, $\mathcal{A}(\mathbf{W}) = V_{4,1}$.

Figure 3: Visual evidence of Theorem 3. a) We construct a BSL having $n = 4$ labels and $d = 2$ features parametrised by $\mathbf{W} \in \mathbb{R}^{4\times 2}$ such that all maximal minors are positive, i.e. $\mathbf{W} \in \mathsf{Gr}^+_{n=4,d=2}$. (b) The rows of the matrix are binary classifiers, we demarcate the decision boundaries for each classifier using a dashed line. (c) We assign each region a sign vector corresponding to which labels the BSL would flag as active for an input falling in that region. As per Theorem 3, exactly the $(d-1) = 1$-alternating outputs are argmaxable. More generally, for $d = 2k+1$, all k-active outputs are argmaxable (see Appendix M).

Definition 4. *For a label assignment* $\mathbf{y}$ *we define* $\mathrm{act}(\mathbf{y})$ *to be the number of active labels in* $\mathbf{y}$*, i.e:*

$$\mathrm{act}(\mathbf{y}) = \sum_{i=1}^{n} \mathbf{1}_{\mathbf{y}}\{\mathbf{y}_i = +\} \tag{4}$$

e.g. $\mathrm{act}(-----) = 0$ *and* $\mathrm{act}(+---+-) = 2$.

Definition 5. *The* k***-active assignments*** *on* n *labels are:*

$$A_{n,k} = \{\mathbf{y} \in \{+,-\}^n : \mathrm{act}(\mathbf{y}) \leq k\} \tag{5}$$

For example, the MIMIC-III dataset (Johnson et al. 2016; Mullenbach et al. 2018) has $n = 8921$ labels, but no example has more than 80 active labels. We now show how to guarantee that all label assignments in $A_{n,k}$ are argmaxable.

4.2 k-Active Argmaxability Guarantees

Our goal in this section is to prove Theorem 4, which states that a *general criterion* for guaranteeing all k-active labels are argmaxable is that the weight matrix $\mathbf{W} \in \mathbb{R}^{n\times d}$, $d \geq 2k+1$ that parametrises the BSL has *maximal minors that agree in sign and are non-zero*. As such, we next introduce maximal minors and the family of matrices with the above property. We prove our result by showing that the argmaxable label assignments for this family of matrices are "$2k$-alternating" and that these subsume the k-active ones.

A **maximal minor** Δ_I of a $n \times d$ matrix, $n > d$, is the determinant of any $d \times d$ submatrix formed by deleting the $n-d$ rows not indexed by I. For example, in Fig. 3a, all maximal minors are positive. We denote with $\mathsf{Gr}^+_{n,d}$ the set of all matrices $\mathbf{W} \in \mathbb{R}^{n\times d}$ whose maximal minors are non-zero and agree in sign (see Appendix C). To prove Theorem 4, we use the following facts known about k-alternating outputs.

Definition 6. *For a label assignment* $\mathbf{y}$ *we define* $\mathrm{alt}(\mathbf{y})$ *to be the number of sign changes encountered when reading the sequence of labels from left to right, i.e:*

$$\mathrm{alt}(\mathbf{y}) = \sum_{i=1}^{n-1} \mathbf{1}_{\mathbf{y}}\{\mathbf{y}_i \neq \mathbf{y}_{i+1}\} \tag{6}$$

e.g: $\mathrm{alt}(++---) = 1$ and $\mathrm{alt}(++-+-) = 3$.

Definition 7. *The* k***-alternating assignments*** *on* n *labels are:*

$$V_{n,k} = \{\mathbf{y} \in \{+,-\}^n : \mathrm{alt}(\mathbf{y}) \leq k\} \tag{7}$$

Lemma 1. $\mathbf{y}$ *is* k*-active* $\implies$ $\mathbf{y}$ *is* $2k$*-alternating.*

See Appendix D.1 for reasoning.

Theorem 2. *(Gantmakher and Kreĭn 1961) see (Karp 2017, Theorem 1.1). If all maximal minors of* $\mathbf{W} \in \mathbb{R}^{n\times d}$ *are non-zero and have the same sign, all label assignments* $\mathbf{y}$ *computed as* $\mathbf{y} = \mathrm{sign}(\mathbf{W}\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$ *are* $d-1$ *alternating.* $\mathbf{W} \in \mathsf{Gr}^+_{n,d} \implies \mathrm{alt}(\mathbf{y}) \leq d-1$.

Theorem 3. *For* $\mathbf{W} \in \mathsf{Gr}^+_{n,d}$ *the argmaxable label assignments are the* $(d-1)$*-alternating vectors.* $\mathbf{W} \in \mathsf{Gr}^+_{n,d} \implies \mathcal{A}(\mathbf{W}) = V_{n,d-1}$.

See Section 4 for intuition and Appendix D.2 for a proof.

Theorem 4. *Consider a BSL parametrised by* $\mathbf{W} \in \mathsf{Gr}^+_{n,2k+1}$ *which predicts label assignments using argmax prediction,* $\mathbf{y} = \mathrm{sign}(\mathbf{W}\mathbf{x})$*. All* k*-active label assignments are argmaxable:* $A_{n,k} \subset \mathcal{A}(\mathbf{W})$.

Proof. From Theorem 3, for $\mathbf{W}$ in $\mathsf{Gr}^+_{n,2k+1}$ the set of $2k$-alternating label assignments $V_{n,2k}$ is argmaxable. Then, from Lemma 1, we have $A_{n,k} \subseteq V_{n,2k} = \mathcal{A}(\mathbf{W})$, and therefore all k-active labels are argmaxable. □

In summary, we have showed that if $\mathbf{W} \in \mathsf{Gr}^+_{n,2k+1}$, all k-active outputs are argmaxable, but we have not given a concrete implementation. We next introduce the DFT layer, a practical and efficient member of the $\mathsf{Gr}^+_{n,2k+1}$ family.

4.3 DFT Layers

Herein we engineer a specific BSL parametrisation that satisfies Theorem 4 and relies on the Discrete Fourier Transform (DFT). In addition to this property, the DFT is also appealing because: a) it allows us to reduce the number

of learnable parameters as the DFT coefficients can be fixed [7] and b) we can compute the activation of the DFT Layer in $\mathcal{O}(n \log n)$ time via the Fast Fourier Transform (see Appendix E) instead of a more expensive $\mathcal{O}(n^2)$ generic matrix-vector product. We next describe a truncated DFT matrix and show that it provides the guarantees we seek. For the DFT matrix, we truncate frequencies larger than k:

$$\boldsymbol{W}^{\mathsf{DFT}}_{n,2k+1} = \begin{bmatrix} \frac{1}{\sqrt{n}} & \sqrt{\frac{2}{n}}\cos t_1 & \sqrt{\frac{2}{n}}\sin t_1 & \cdots & \sqrt{\frac{2}{n}}\cos kt_1 & \sqrt{\frac{2}{n}}\sin kt_1 \\ \frac{1}{\sqrt{n}} & \sqrt{\frac{2}{n}}\cos t_2 & \sqrt{\frac{2}{n}}\sin t_2 & \cdots & \sqrt{\frac{2}{n}}\cos kt_2 & \sqrt{\frac{2}{n}}\sin kt_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{\sqrt{n}} & \sqrt{\frac{2}{n}}\cos t_n & \sqrt{\frac{2}{n}}\sin t_n & \cdots & \sqrt{\frac{2}{n}}\cos kt_n & \sqrt{\frac{2}{n}}\sin kt_n \end{bmatrix},$$
$$t_i = \frac{2\pi(i-1)}{n},\ i \in [n] \tag{8}$$

Lemma 2. *A truncated DFT matrix* $\boldsymbol{W}^{\mathsf{DFT}}_{n,2k+1} \in \mathsf{Gr}^{+}_{n,2k+1}$.

We need to show that the maximal minors of $\boldsymbol{W}^{\mathsf{DFT}}_{n,2k+1}$ are non-zero and agree in sign. See Appendix D.3 for a proof.

Problem: Regions can become too small. While in practice we could use the fixed DFT Layer as defined above and rely on an expressive feature encoder to do the heavy lifting, if the number of labels, n, is much greater than the number of features, $2k+1$, it becomes hard to classify some label assignments with large confidence. This is because segmenting a low dimensional space with very many hyperplanes induces regions that become arbitrarily small shards. In argmaxability terms, if we fix an ϵ and increase the number of labels, all k-active labels are argmaxable but increasingly more are not ϵ-argmaxable, see left in Fig. 4. This is a problem for any classifier $\mathbf{W}$, because for training and for generalisation we need to project points into large enough regions. However, we found that DFT layers are more susceptible to it (Appendix F.2) than general BSL (see Appendix I for a more detailed explanation).

Solution: Slack variables. A practical way to deal with small regions is to increase the dimensionality of the features by adding learnable *slack variables*, see Fig. 4. Crucially, when doing so we retain our guarantees, as we show below.

Lemma 3. *Assume a label assignment* $\mathbf{y}$ *is argmaxable for a classifier* $\mathbf{W} \in \mathbb{R}^{n \times d}$. *Consider increasing the dimensionality of the features of the classifier* $\mathbf{W}$ *by adding* s *more randomly initialised slack columns* $\mathbf{S} \in \mathbb{R}^{n \times s}$. *Then* $\mathbf{y}$ *is also argmaxable in* $\mathbf{W}' = [\mathbf{W}\ \mathbf{S}]$, $\mathbf{W}' \in \mathbb{R}^{n \times (d+s)}$.

Proof. Consider the input feature vector for $\mathbf{W}'$, $\mathbf{x}' = \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_s \end{bmatrix}$, $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{x}_s \in \mathbb{R}^s$. Set $\mathbf{x}_s = \mathbf{0}$. Then notice that $\mathbf{y} = \operatorname{sign}\left([\mathbf{W}\ \mathbf{S}]\begin{bmatrix} \mathbf{x} \\ \mathbf{0} \end{bmatrix}\right) = \operatorname{sign}(\mathbf{W}\mathbf{x})$ is equivalent to the original classifier, so if $\mathbf{y}$ is argmaxable in $\mathbf{W}$ it is also argmaxable in $\mathbf{W}'$ by setting $\mathbf{x}_s$ to zero. □

As such, we propose the DFT layer, which has $(2k+1) \times n$ fixed parameters which enforce argmaxability and $s \times n$ learnable parameters which give it flexibility.

[7] In early experiments we parametrised and learned the t_i of the DFT matrix but found little impact on the results.

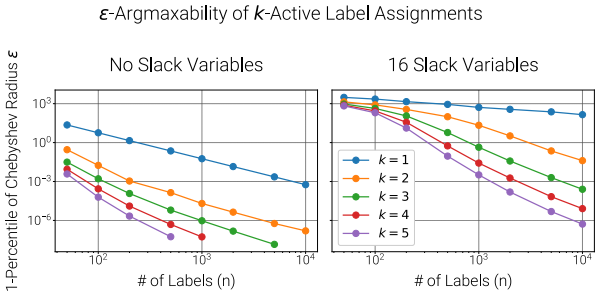


Figure 4: Left: As we increase the number of labels n for the DFT Layer, the radii of the regions shrink, making them harder to predict in practice. Right: Adding slack variables ameliorates this problem. We plot ϵ-argmaxability (Definition 3), measured here for the 1% of labels that have radius less than that plotted. For the DFT Layer, i.e. $\mathbf{W} = \boldsymbol{W}^{\mathsf{DFT}}_{n,2k+1}$, all k-active label assignments are argmaxable, but as we increase n, some (see $k \geq 3$) cannot be detected at the precision of the LP (10^{-8}). Adding 16 randomly initialised slack columns, i.e. $\mathbf{W} = \left[\boldsymbol{W}^{\mathsf{DFT}}_{n,2k+1}\ \mathbf{S}\right]$, makes the regions ϵ-argmaxable with larger ϵ.

5 Experiments

We now empirically evaluate the BSL and DFT layers on three MLC datasets and answer the following research questions: **RQ1)** Do BSLs have unargmaxable labels in practice? **RQ2)** Can DFT layers guarantee that meaningful labels are argmaxable in practice? **RQ3)** What is the trade-off between performance and the number of trainable parameters? We answer the above after introducing the models and datasets.

5.1 Model Setup

We define the two MLC output layers we will compare, the *BSL Layer* that is unconstrained and does not guarantee argmaxability of k-active outputs, and our *DFT layer* which does. In our experiments we want to study the effect of varying the bottleneck width irrespective of the feature dimensionality of the encoder which varies across datasets and models. As such, we introduce an *affine projection layer* (parametrised by $\mathbf{P}$ and $\mathbf{b}$) between the feature encoder and the *linear classifier* (parametrised by $\mathbf{W}$). For simplicity, we do not include bias terms for the classifiers. For both models, we compute the logits $\mathbf{z} \in \mathbb{R}^n$ from the encoder activation $\mathbf{e} \in \mathbb{R}^e$ as:

$$\mathbf{z} = \mathbf{W}\mathbf{x}, \quad \mathbf{x} = \mathbf{P}\mathbf{e} + \mathbf{b} \tag{9}$$

While the classifiers $\mathbf{W}$ have the same number of learnable parameters for both output layers, the parametrisations differ, as we discuss next.

BSL Output Layer For the BSL, the projection layer maps from e, the dimensionality of the encoder embeddings, to d, the feature dimensionality using $\mathbf{P} \in \mathbb{R}^{e \times d}$ and bias $\mathbf{b} \in \mathbb{R}^d$. This is followed by the linear classifier $\mathbf{W} \in \mathbb{R}^{n \times d}$.

DFT Output Layer For the DFT, we first pick the maximum number of active labels, k, depending on the statis-

tics of the dataset. We then set the number of slack dimensions to be d so we can directly compare to the BSL. As such, we have $\mathbf{P} \in \mathbb{R}^{e\times(2k+1+d)}$ and $\mathbf{b} \in \mathbb{R}^{(2k+1+d)}$, since we include $2k+1$ more features that map to the DFT columns of the classifier. The learnable parameters of the classifier comprise d slack columns. Conceptually, and for the purposes of checking the classifier with our LP, we construct the classifier by concatenating the fixed DFT matrix to the slack columns, i.e. $\mathbf{W} = \left[\boldsymbol{W}^{\mathsf{DFT}}_{n,2k+1}\ \mathbf{S}\right]$, $\mathbf{W} \in \mathbb{R}^{n\times(2k+1+d)}$. In practice we compute the logits $\mathbf{z}$ efficiently as $\mathbf{z} = \mathrm{FFT}(\mathbf{x}_{:2k+1}) + \mathbf{S}\mathbf{x}_{2k+1:}$ (see Appendix E).

Computational Cost of DFT Compared to the BSL, the cost of the DFT layer with n labels is a) an additional cheap $\mathcal{O}(n \log n)$ matrix vector multiplication and b) an additional $e \times (2k+1)$ trainable parameters in the projection layer. However, for models with large n, we can easily offset the increase in parameters if we want, by modestly shrinking the slack d of the DFT output layer. For example, the MIMIC-III CNN models (Mullenbach et al. 2018) have $n = 8921$ and $e = 500$. For $k = 80$, a DFT adds 500×161 parameters to the projection layer. We could offset this by decreasing d in the output layer by only $\lceil \frac{500\times161}{8921} \rceil = 10$. In Section 5.3 we show that DFT layers obtain better performance with lower d than BSLs, and as such can be more parameter efficient.

Faster Training of DFT For the DFT, we introduce an initialisation trick to speed up training. We exploit that a) $\mathbf{W}^{\mathsf{DFT}}$ is known and fixed and b) the outputs are k-active. Since the outputs are k-active, we would prefer to assign a probability $\frac{k}{n}$ to all labels when we start training. To achieve this, we can exploit the fact that the first column of $\mathbf{W}^{\mathsf{DFT}}$ is $\frac{1}{\sqrt{n}}$ and initialise the bias vector of the projection layer to be $[\sqrt{n}\,\mathrm{logit}(\frac{k}{n}), 0, \ldots, 0]$, where the logit function is the inverse of sigmoid. This way, assuming logits are close to zero when we begin training, the model will assign probability $\approx \frac{k}{n}$ to each label instead of $\approx \frac{1}{2}$. A similar bias initialisation idea for MLC was discussed in (Schultheis and Babbar 2022), but it was not used in a neural network.

5.2 MLC Tasks (Datasets)

We now introduce the three datasets. We summarise their important attributes in Table 1. See Appendix H for details on dataset construction and more dataset statistics.

Clinical Coding (MIMIC-III) We first test the DFT layer on MIMIC-III (Johnson et al. 2016). For this safety critical application of clinical coding, the goal is to tag each clinical note with a set of relevant ICD-9 codes which describe findings (see Fig. 1). We retrain the CNN encoder model defined in Mullenbach et al. (2018) which has $n = 8921$ and $e = 500$. We use the same word embeddings, preprocessed data, data splits, metrics (Prec@8) and hyperparameters reported in the paper (Mullenbach et al. 2018). We only change the learning rate of the Adam optimiser to 0.001, as this improves results (as also found by Edin et al. (2023)).

MLC Dataset	n	k	N	encoder
MIMIC-III	8921	80	44k	CNN (e=500)
BioASQ task A	20000	50	100k	BERT (e=768)
OpenImagesV6	8933	50	108k	TResnet (e=2432)

Table 1: Setup: Number of labels, n, max number of active labels, k, and number of training examples, N.

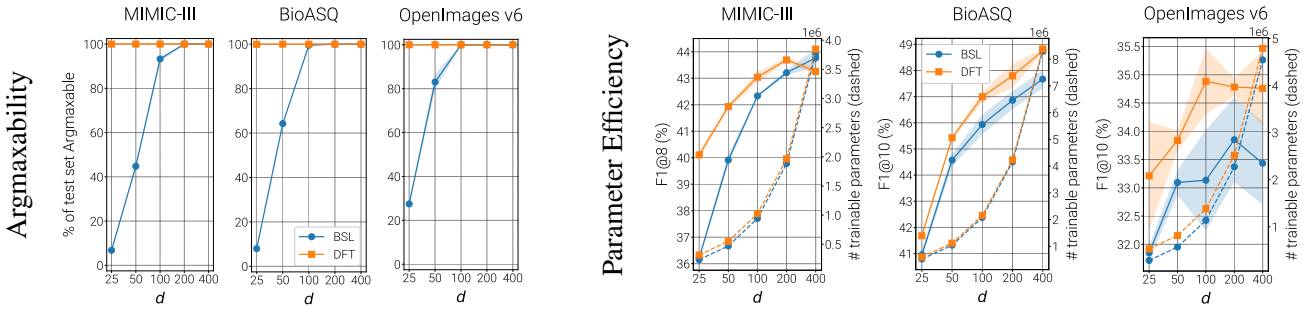
Semantic Indexing (BioASQ Task A) Next, we focus on the 2021 BioASQ semantic indexing challenge (Tsatsaronis et al. 2015; Nentidis et al. 2021; Krithara et al. 2023). For this task, we are given PubMed abstracts and asked to predict a set of relevant MeSH headings[8] for each article. We create dataset splits (see Appendix H.2 for details) with $n = 20000$, making sure that all individual labels occur in both the train and test sets. We do so to avoid claiming a label assignment is unargmaxable when in fact it would be hard to predict the labels that constitute it. We use $k = 50$ as this is the maximum number of active labels per example for this dataset by construction. We finetune PubMedBERT (Gu et al. 2021), a domain specific uncased BERT (Devlin et al. 2019) encoder that has been pretrained on PubMed abstracts and has $e = 768$. We use early stopping with a patience of 10 on the validation crossentropy loss.

Image MLC (OpenImages v6) We use the OpenImages v6 dataset (Kuznetsova et al. 2020) where the goal is to tag each image with objects that appear in it. Similar to the BioASQ case, we choose the label vocabulary $n = 8933$ such that all examples in the train, validation and test set are covered. We pick $k = 50$ since the training data has at most 45 active labels per example. We finetune the TResnet (Ridnik et al. 2020) with $e = 2432$ that Baruch et al. (2020) pretrained for MLC on this dataset. We use early stopping with a patience of 10 on the validation crossentropy loss.

5.3 Results

RQ1) BSL: Unargmaxable outputs. We use the LP to verify the BSL on the test sets. As can be seen in Fig. 5a, for $d > 200$ all the examples in the test set are argmaxable for the BSL. However, as we reduce d, the number of unargmaxable label assignments increases for all datasets. More specifically, we first get unargmaxable outputs at $d = 200$ for MIMIC-III (see Appendix F). Analogous considerations can be drawn for BioASQ and OpenImages (Fig. 5a). As such, we conclude that unargmaxability can indeed be an issue when d is not large enough. Note that one can never determine a d that can always guarantee all label configurations of interest to be argmaxable: even an exhaustive verification on all test samples does not imply that future unseen configurations will be argmaxable. **RQ2) DFT: Argmaxability.** While we know that the DFT layer guarantees argmaxability in theory, we also see that it works in practice (apart from a handful of outputs when $d = 25$, which are ϵ-unargmaxable, see Appendix F.2). Crucially, these guarantees also apply to unseen k-active label assignments; this would be impossible to enforce with BSLs, as discussed. **RQ3) Parameter Efficiency.** In addition, we turn to Fig. 5b, and see that the DFT layer outperforms the BSL layer by a wide margin for small

[8] https://www.nlm.nih.gov/mesh/meshhome.html



(a) Our DFT layer does not suffer from unargmaxable test examples. BSLs cannot provide any such guarantees.

(b) Our DFT layer is more parameter efficient. F1 is overall better than the BSL, so we can match BSL's F1 with fewer trainable parameters (smaller d).

Figure 5: Comparison of BSL and our DFT output layer on three MLC datasets. We vary, d, the number of trainable dimensions and plot the mean and std (shaded) over 3 runs with different random seeds. Left: We use the LP to verify the output layers on the test sets. When $d < 200$ a large percentage of label configurations becomes unargmaxable for the BSL, in contrast to our DFT. Right: Performance in F1@8 or F1@10 (left axis) in terms of the number of trainable parameters (right axis, dashed lines). Our DFT is better ($d \leq 200$) or comparable to the BSL (MIMIC-III, $d > 200$). We can therefore retain a high F1@k score for DFT even if we reduce the number of trainable parameters by making the bottleneck narrower. For example, on BioASQ we can surpass the F1@10 of the BSL that has $d = 200$ with a DFT of $d = 100$, which has about 50% fewer trainable parameters.

d. This allows us to match the performance of the BSL using a DFT with smaller d and hence fewer trainable parameters. As can be seen, in some cases DFT layers obtain better or comparable performance with up to 50% less trainable parameters. **Faster Training.** Lastly, a benefit of DFT layers is that they converge faster due to the initialisation trick (Section 5.1), see Appendix G for a comparison of training times.

6 Related Work

Limitations of Low-rank Parametrisations in MLC. Previous work has shown that the low-rank assumption can be problematic for MLC: long-tail labels lack strong linear dependencies, making the output label matrix high-rank (Bhatia et al. 2015; Xu, Tao, and Xu 2016; Tagami 2017). As such, low-rank approximations suffer from high reconstruction error. Herein, we highlight a more tangible limitation: meaningful label assignments can be unargmaxable. While we prove that we can retain argmaxability with a very low-rank matrix if the outputs are sparse, we concur that low-rank parametrisations can still be problematic, especially for predicting long-tail labels. For effective prediction of such labels, Babbar and Schölkopf (2019) showed the importance of making a model robust to input perturbations. We similarly find that we need ϵ-argmaxability with a large enough ϵ, both for effective training and prediction.

Sign Rank of a Matrix. Consider approximating a matrix of sign vectors with a matrix $\mathbf{M}$ of rank r. The smallest r for which all sign vectors can be recovered by taking the sign of $\mathbf{M}$ is known as its sign rank (Alon, Moran, and Yehudayoff 2015; Neumann, Gemulla, and Miettinen 2016), see Appendix L for more details. Our rank $2k + 1$ approximation of a k-active label matrix is a non-trivial upper bound on its sign rank, see also Alon, Frankl, and Rodl (1985, Theorem 1.1, part i). Chanpuriya et al. (2020, Theorem 6) use a similar construction to show that adjacency matrices of graphs of bounded degree k can be embedded in $2k+1$ dimensions.

7 Discussion and Conclusion

Through extensive experiments on three multi-label classification datasets we have shown that BSLs – which are still ubiquitous in large MLC scenarios (Section 2) – can have unargmaxable label assignments when the label vocabulary is much larger than the number of input features (Section 5). We believe practitioners should be aware that selecting the feature dimension of a BSL is not an innocuous hyperparameter search: by lowering the rank of the output layer they are potentially making some label assignments impossible to predict. Unargmaxability impacts the generalisation and trustworthiness of these classifiers, since BSLs cannot guarantee that meaningful label assignments will not be missed at test time or that the classifiers will not be targetted by adversarial attacks (Zhu et al. 2019; Aghakhani et al. 2021).

However, we showed that this does *not* have to be the case. We provided a family of parametrisations that guarantee that all label assignments having up to k active labels are argmaxable (Section 4.2) and implemented our DFT output layer which converges faster than a BSL, is up to 50% more parameter efficient, and outperforms or matches the BSL on three widely used MLC datasets.

Our findings also prompt several avenues for future work. As we make the sigmoid bottleneck narrower, the argmaxable regions get smaller (Fig. 4), making label assignments harder to predict robustly. Can we parametrise output layers in a way that guarantees ϵ-argmaxability for large ϵ? Moreover, while we focussed on BSLs, there are many families of output layers to analyse. E.g. those that a) partition the label vocabulary so that classifiers do not all compete in a shared feature space (Yu, Zhong, and Dhillon 2020), b) parametrise the classifier based on the input (Mullenbach et al. 2018) and c) predict labels autoregressively (Simig et al. 2022; Kementchedjhieva and Chalkidis 2023). It is an open question for future work to verify that these models do not admit unargmaxable outputs and to robustify them otherwise.

Acknowledgements

We would like to thank Lorenzo Loconte, Rickey Liang, and Marina Potsi for feedback and Emile van Krieken for feedback and catching omissions. Furthermore, Asif Khan for many fruitful discussions and help connecting the Cyclic Polytope to truncated DFT, Nikolay Bogoychev for discussing multi-label classification datasets, Jesse Sigal for discussions on cyclic permutations, Matúš Falis for discussing multi-label classification and datasets, Tom Sherbourne for discussing DFT and low-pass filters and Ivan Titov and Henry Gouk for helpful suggestions and advice.

AG was supported by the Engineering and Physical Sciences Research Council (EP/R513209/1). AV was supported by the "UNREAL: Unified Reasoning Layer for Trustworthy ML" project (EP/Y023838/1) selected by the ERC and funded by UKRI EPSRC.

References

Aghakhani, H.; Meng, D.; Wang, Y.-X.; Kruegel, C.; and Vigna, G. 2021. Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, 159–178. IEEE.

Alon, N.; Frankl, P.; and Rodl, V. 1985. Geometrical realization of set systems and probabilistic communication complexity. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, 277–280.

Alon, N.; Moran, S.; and Yehudayoff, A. 2015. Sign rank versus VC dimension. In *Annual Conference Computational Learning Theory*.

Babbar, R.; and Schölkopf, B. 2019. Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, 1–23.

Baruch, E. B.; Ridnik, T.; Zamir, N.; Noy, A.; Friedman, I.; Protter, M.; and Zelnik-Manor, L. 2020. Asymmetric Loss For Multi-Label Classification. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 82–91.

Bhatia, K.; Jain, H.; Kar, P.; Varma, M.; and Jain, P. 2015. Sparse Local Embeddings for Extreme Multi-label Classification. In Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Björner, A.; Las Vergnas, M.; Sturmfels, B.; White, N.; and Ziegler, G. M. 1999. *Oriented Matroids*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, second edition.

Boyd, S. P.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge University Press.

Chanpuriya, S.; Musco, C.; Sotiropoulos, K.; and Tsourakakis, C. 2020. Node Embeddings and Exact Low-Rank Representations of Complex Networks. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 13185–13198. Curran Associates, Inc.

Choi, E.; Levy, O.; Choi, Y.; and Zettlemoyer, L. 2018. Ultra-Fine Entity Typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 87–96. Melbourne, Australia: Association for Computational Linguistics.

Cordovil, R.; and Duchet, P. 2000. Cyclic Polytopes and Oriented Matroids. *European Journal of Combinatorics*, 21(1): 49–64.

Cover, T. M. 1965. Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *IEEE Transactions on Electronic Computers*, EC-14(3): 326–334.

Demeter, D.; Kimmel, G.; and Downey, D. 2020. Stolen Probability: A Structural Weakness of Neural Language Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2191–2197. Online: Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Donoho, D. L.; and Tanner, J. 2005. Sparse nonnegative solution of underdetermined linear equations by linear programming. *Proceedings of the National Academy of Sciences of the United States of America*, 102 27: 9446–51.

Edin, J.; Junge, A.; Havtorn, J. D.; Borgholt, L.; Maistro, M.; Ruotsalo, T.; and Maaløe, L. 2023. Automated Medical Coding on MIMIC-III and MIMIC-IV: A Critical Review and Replicability Study. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Gale, D. 1963. Neighborly and cyclic polytopes. In *Proceedings of Symposia in Pure Mathematics*, volume 7, 225–232.

Ganea, O.; Gelly, S.; Becigneul, G.; and Severyn, A. 2019. Breaking the Softmax Bottleneck via Learnable Monotonic Pointwise Non-linearities. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 2073–2082. PMLR.

Gantmakher, F.; and Kreĭn, M. 1961. *Oscillation Matrices and Kernels and Small Vibrations of Mechanical Systems*. American Mathematical Society. ISBN 9780821831717.

Grivas, A.; Bogoychev, N.; and Lopez, A. 2022. Low-Rank Softmax Can Have Unargmaxable Classes in Theory but Rarely in Practice. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 6738–6758. Dublin, Ireland: Association for Computational Linguistics.

Gu, Y.; Tinn, R.; Cheng, H.; Lucas, M.; Usuyama, N.; Liu, X.; Naumann, T.; Gao, J.; and Poon, H. 2021. Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. *ACM Trans. Comput. Healthcare*, 3(1).

Gurobi Optimization. 2021. Gurobi Optimizer Reference Manual.

Jain, H.; Balasubramanian, V.; Chunduri, B.; and Varma, M. 2019. Slice: Scalable Linear Extreme Classifiers Trained on 100 Million Labels for Related Searches. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*.

Järvelin, K.; and Kekäläinen, J. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20: 422–446.

Johnson, A. E. W.; Pollard, T. J.; Shen, L.; Lehman, L.-W. H.; Feng, M.; Ghassemi, M.; Moody, B.; Szolovits, P.; Celi, L. A.; and Mark, R. G. 2016. MIMIC-III, a freely accessible critical care database. *Sci. Data*, 3(1): 160035.

Karp, S. N. 2017. Sign variation, the Grassmannian, and total positivity. *Journal of Combinatorial Theory, Series A*, 145: 308–339.

Kementchedjhieva, Y.; and Chalkidis, I. 2023. An Exploration of Encoder-Decoder Approaches to Multi-Label Classification for Legal and Biomedical Text. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Findings of the Association for Computational Linguistics: ACL 2023*, 5828–5843. Toronto, Canada: Association for Computational Linguistics.

Krithara, A.; Mork, J. G.; Nentidis, A.; and Paliouras, G. 2023. The road from manual to automatic semantic indexing of biomedical literature: a 10 years journey. *Frontiers in Research Metrics and Analytics*, 8.

Kulmanov, M.; and Hoehndorf, R. 2019. DeepGOPlus: improved protein function prediction from sequence. *Bioinformatics*, 36: 422 – 429.

Kuznetsova, A.; Rom, H.; Alldrin, N.; Uijlings, J.; Krasin, I.; Pont-Tuset, J.; Kamali, S.; Popov, S.; Malloci, M.; Kolesnikov, A.; Duerig, T.; and Ferrari, V. 2020. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*.

Mullenbach, J.; Wiegreffe, S.; Duke, J. D.; Sun, J.; and Eisenstein, J. 2018. Explainable Prediction of Medical Codes from Clinical Text. In *NAACL*.

Nentidis, A.; Katsimpras, G.; Vandorou, E.; Krithara, A.; and Paliouras, G. 2021. Overview of BioASQ Tasks 9a, 9b and Synergy in CLEF2021. In *Conference and Labs of the Evaluation Forum*.

Neumann, S.; Gemulla, R.; and Miettinen, P. 2016. What You Will Gain By Rounding: Theory and Algorithms for Rounding Rank. *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 380–389.

Paul, S.; and Rakshit, S. 2014. Large-scale multi-label text classification. https://keras.io/examples/nlp/multi_label_classification. Accessed: 2023-08-15.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Pinkus, A. 2009. *Basic properties of totally positive and strictly totally positive matrices*, 1–35. Cambridge Tracts in Mathematics. Cambridge University Press.

Postnikov, A. 2006. Total positivity, Grassmannians, and networks. *arXiv: Combinatorics*.

Ridnik, T.; Lawen, H.; Noy, A.; and Friedman, I. 2020. TResNet: High Performance GPU-Dedicated Architecture. *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1399–1408.

Schultheis, E.; and Babbar, R. 2022. Speeding-up one-versus-all training for extreme classification via mean-separating initialization. *Machine Learning*, 111(11): 3953–3976.

Simig, D.; Petroni, F.; Yanki, P.; Popat, K.; Du, C.; Riedel, S.; and Yazdani, M. 2022. Open Vocabulary Extreme Classification Using Generative Models. In *Findings of the Association for Computational Linguistics: ACL 2022*, 1561–1583. Dublin, Ireland: Association for Computational Linguistics.

Sturmfels, B. 1988. Neighborly Polytopes and Oriented Matroids. *European Journal of Combinatorics*, 9(6): 537–546.

Tagami, Y. 2017. AnnexML: Approximate Nearest Neighbor Search for Extreme Multi-label Classification. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Tsatsaronis, G.; Balikas, G.; Malakasiotis, P.; Partalas, I.; Zschunke, M.; Alvers, M. R.; Weissenborn, D.; Krithara, A.; Petridis, S.; Polychronopoulos, D.; Almirantis, Y.; Pavlopoulos, J.; Baskiotis, N.; Gallinari, P.; Artières, T.; Ngomo, A.-C. N.; Heino, N.; Gaussier, É.; Barrio-Alvers, L.; Schroeder, M.; Androutsopoulos, I.; and Paliouras, G. 2015. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16.

Xu, C.; Tao, D.; and Xu, C. 2016. Robust Extreme Multi-label Learning. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Yang, Z.; Dai, Z.; Salakhutdinov, R.; and Cohen, W. W. 2018. Breaking the Softmax Bottleneck: A High-Rank RNN Language Model. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Yu, H.-F.; Zhong, K.; and Dhillon, I. S. 2020. PECOS: Prediction for Enormous and Correlated Output Spaces. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Zhu, C.; Huang, W. R.; Li, H.; Taylor, G.; Studer, C.; and Goldstein, T. 2019. Transferable clean-label poisoning attacks on deep neural nets. In *International Conference on Machine Learning*, 7614–7623. PMLR.

Ziegler, G. M. 1994. *Lectures on Polytopes*. Springer New York, NY.