Check for updates

Multi-target prediction for dummies using two-branch neural networks

Dimitrios Iliadis[1] [ID] · **Bernard De Baets**[1] · **Willem Waegeman**[1]

© The Author(s), under exclusive licence to Springer Science+Business Media LLC, part of Springer Nature 2021

Abstract

Multi-target prediction (MTP) serves as an umbrella term for machine learning tasks that concern the simultaneous prediction of multiple target variables. Classical instantiations are multi-label classification, multivariate regression, multi-task learning, dyadic prediction, zero-shot learning, network inference, and matrix completion. Despite the significant similarities, all these domains have evolved separately into distinct research areas over the last two decades. This led to the development of a plethora of highly-engineered methods, and created a substantially-high entrance barrier for machine learning practitioners that are not experts in the field. In this work we present a generic deep learning methodology that can be used for a wide range of multi-target prediction problems. We introduce a flexible multi-branch neural network architecture, partially configured via a questionnaire that helps end users to select a suitable MTP problem setting for their needs. Experimental results for a wide range of domains illustrate that the proposed methodology manifests a competitive performance compared to methods from specific MTP domains.

Keywords Multi-target prediction · Multi-label classification · Multivariate regression · Matrix completion · Multi-task learning · Dyadic prediction

✉ Dimitrios Iliadis
dimitrios.iliadis@ugent.be

Bernard De Baets
bernard.debaets@UGent.be

Willem Waegeman
willem.waegeman@ugent.be

[1] KERMIT, Department of Data Analysis and Mathematical Modelling, Ghent University, Coupure links 653, 9000 Ghent, Belgium

1 Introduction

Over the last decade, multi-target prediction (MTP) has emerged as a novel umbrella term, unifying supervised learning techniques that are concerned with predicting multiple target variables at the same time. In principle, these targets can be of different types, such as nominal, ordinal, or real-valued. Driven by tutorials and workshops at international conferences, such as ICML 2013 and ECML/PKDD 2014, 2015 and 2018, the area of MTP has attracted significant interest in the machine learning community. Its applicability potential is continuously increasing, as more and more real-world problems require the simultaneous prediction of multiple targets.

In the field of machine learning one can identify many classical examples of MTP tasks, such as the image tagging task from the area of computer vision (Wang et al. 2016; Wei et al. 2015; Yan et al. 2019), the document tagging task from the field of text mining (Chen et al. 2017; Huang et al. 2019), as well as the product recommendation task that is prevalent in online retailing (Fu et al. 2018; Wei et al. 2017). In addition to these typical examples, one can also identify instances of MTP-related applications that are less well known yet important. In the field of climate science, forecasting the weather in different areas of the world at the same time is a quite complicated task that necessitates the modeling of relationships between various atmospheric processes (Papagiannopoulou et al. 2018). In medicine, patients can usually be associated with multiple interacting pathologies at the same time (Baltruschat et al. 2019; Kumar et al. 2018; Chen et al. 2019a). Finally, the emergence of the latest pandemic has highlighted the importance of rapid drug discovery (Pliakos et al. 2019; Rifaioglu et al. 2020; Jin et al. 2017). In this field, the initial goal is to find a set of chemical compounds that show high binding affinity with a biological target, so the use of automated multi-target prediction methods can provide a much-needed speedup.

All these applications are usually encountered in machine learning papers as use cases for specialized techniques. These techniques typically belong to well-known subfields like multi-label classification (Yeh et al. 2017; Read et al. 2009; Tsoumakas et al. 2010; Yu et al. 2014; Rokach et al. 2014), multivariate regression (De'Ath 2002; Du and Xu 2017; Xu et al. 2013), multi-task learning (Sener and Koltun 2018; Misra et al. 2016; Liu et al. 2019), dyadic prediction (Menon and Elkan 2011, 2010; Schäfer and Hüllermeier 2015), hierarchical multi-label classification (Wehrmann et al. 2018; Cerri et al. 2014), zero-shot learning (Romera-Paredes and Torr 2015; Norouzi et al. 2013), matrix completion (Jain et al. 2013; Shan and Banerjee 2010), and hybrid matrix completion (Strub et al. 2016; Dong et al. 2017), which from a distance all look quite different from one another. A recent survey (Waegeman et al. 2019) reviewed not less than 100 methods from these subfields from a general multi-target prediction perspective. In addition, a formal mathematical framework to gather those subfields under a single umbrella was expounded.

The said mathematical framework will be the point of departure for the goal of the present paper, which is the development of a general deep learning methodology for multi-target prediction problems. Instead of introducing a method that achieves state-of-the-art performance for a narrow range of problems, we present a flexible two-branch neural network architecture that is applicable in a wide range of MTP problems. This type of architecture shows some resemblance with a few deep learning methods that have been recently proposed for specific tasks, such as collaborative filtering (He et al. 2017; Wang et al. 2019) and metric learning (Hoffer and Ailon 2015; Yi et al. 2014; Mueller and Thyagarajan 2016). However, we are the first to make this architecture

generally accessible for a wide range of multi-target prediction problems. We make the methodology user friendly by introducing a small questionnaire that supports a semi-automated configuration of the two-branch neural network by means of small modifications in its architecture, loss function and inputs. In this way we unlock multi-target prediction to a wide range of users with basic machine learning expertise.

Our can see some parallels between our work and an ongoing trend in deep learning research towards the development of general-purpose neural network architectures instead of architectures that are only useful for a specific problem setting. For example, the chapter on recurrent and recursive nets in the book of Goodfellow et al. (2016) discusses general deep learning architectures for sequence modelling tasks, of which one-to-one, one-to-many, and many-to-many architectures of equal or different length are specific instantiations. Other well-known examples of general-purpose machine learning methodologies are structured support vector machines (Wang et al. 2009; Zhang and Gales 2011), conditional random fields (Lafferty et al. 2001; Zheng et al. 2015) and probabilistic graphical models (Frey and Jojic 2005). Especially in statistics it is very common to develop general-purpose frameworks, see e.g. generalized linear models (McCullagh and Nelder 2019). Those models can be applied to various types of supervised learning problems, such as binary and multi-class classification problems, as well as regression problems involving real-valued, ordinal and count-based targets.

This paper is organized as follows. Section 2 quickly reviews the mathematical framework of Waegeman et al. (2019), which unifies a wide range of multi-target prediction problems. That section also discusses the inner workings of our proposed questionnaire. Section 3 explains several examples of real-world tasks and details how the questionnaire can help with selecting the most suitable MTP problem setting. Section 4 presents a detailed view of the two-branch neural network architecture while emphasizing the main characteristics of its flexibility. Section 5 gives a summary of closely-related work. Section 6 showcases that the proposed methodology works well for a wide range of problems, by comparisons with 15 different methods on 21 different datasets, across 6 MTP problem settings. In the last section, we formulate a conclusion and some future perspectives, discussing the current limitations of our work.

2 Towards a rule-based system for MTP problem setting selection

In this section we introduce the MTP framework, as well as the novel questionnaire we designed in order to identify the proper problem setting. We also detail the four validation settings that are used in the area of MTP.

2.1 The MTP prediction framework

Let us start with the formal definition of a multi-target prediction problem, as introduced in Waegeman et al. (2019).

Definition 1 A multi-target prediction problem is characterized by instances $\mathbf{x} \in \mathcal{X}$ and targets $\mathbf{t} \in \mathcal{T}$ with the following properties:

(P1) A training dataset $\mathcal{D}$ is comprised of triplets $(\mathbf{x}_i, \mathbf{t}_j, y_{ij})$, where $\mathbf{x}_i$ represents an instance ($i \in \{1, \ldots, n\}$), $\mathbf{t}_j$ represents a target ($j \in \{1, \ldots, m\}$), and $y_{ij} \in \mathcal{Y}$ is the score that quantifies their relationship. This dataset can be arranged in an $n \times m$ matrix $\mathbf{Y}$ that is usually incomplete.

(P2) The score set $\mathcal{Y}$ consists of nominal, ordinal or real values.

(P3) The objective is to predict the score for any instance-target couple $(\mathbf{x}, \mathbf{t}) \in \mathcal{X} \times \mathcal{T}$.

Intentionally, this definition is kept very general in order to cover a wide range of MTP settings. In Waegeman et al. (2019) also formal definitions are given for the most common settings, grouped into three categories:

- MTP settings without any kind of usable features (side information) for the targets: this includes the more conventional settings, such as multi-label classification, multivariate regression and multi-task learning.
- MTP settings with side information for targets: this includes settings such as hierarchical multi-label classification, dyadic prediction, multi-task learning with task features, zero-shot learning and matrix completion with side information.
- Non-MTP settings: these are settings that could be expressed as multi-target prediction settings, but are not covered for technical reasons. Two such cases are multi-class classification and structured output prediction.

We do not repeat all those definitions here, but refer the interested reader to Appendix A. However, going over the various definitions is not unimportant in view of understanding the purpose of the questionnaire that is introduced next. So, let us see what we get for by far the most popular setting in literature, namely multi-label classification.

Definition 2 The **multi-label classification** setting is an instance of the MTP framework with the following additional properties:

(P4) All targets are observed during training ($|\mathcal{T}| = m$).

(P5) No side information is available for targets, thus we identify them with natural numbers ($\mathbf{t}_j = j$).

(P6) The score matrix $\mathbf{Y}$ is fully observed.

(P7) The score set is $\mathcal{Y} = \{0, 1\}$.

One can see that for multi-label classification three additional properties appear, in addition to the four general properties that hold for all MTP problems. In Appendix A we provide similar definitions for multivariate regression, multi-task learning, hierarchical multi-label classification, dyadic prediction, zero-shot learning, and matrix completion with and without side information. All those settings have some specific properties, and the purpose of the questionnaire will be to map the answers of users to such properties.

2.2 The rule-based system

We are able to propose the appropriate MTP problem setting using a rule-based system deployed on-top of a purpose-built questionnaire. The questionnaire is partly answered automatically with our framework from the characteristics of the dataset. There are also

questions that currently can only be answered by the user and that have been carefully designed to extract his/her intentions about the given problem. We imagine that by using a graphical interface that accepts the test set, a future version can automatically detect whether the user expects a generalization to unseen instances or targets. In the current stage of development, we use the following questions:

Q1: Is it expected to encounter novel instances during testing? **(yes/no)**
Q2: Is it expected to encounter novel targets during testing? **(yes/no)**
Q3: Is there side information available for the instances? **(yes/no)**
Q4: Is there side information available for the targets? **(yes/no)**
Q5: Is the score matrix fully observed? **(yes/no)**
Q6: What is the type of the target variable? **(binary/nominal/ordinal/real-valued)**

These questions are designed to determine the possibility of encountering novel instances or targets during the test phase, the availability of usable side information in the form of relations or representations for instances and targets, the sparsity of the score matrix and the type of values inside the matrix. The aforementioned questions generate 128 different combinations. We have internally annotated the most popular cases with the appropriate multi-target prediction setting (see Table 1), thus transferring our expert knowledge into the rule-based system. There are, however, some specific combinations of characteristics that make the resulting example unable to be annotated. These examples usually try to generalize to novel instances or targets without providing the appropriate side information.

The mentioned differences in the availability of side information that is traditionally associated with each MTP problem setting has led to the distinction of several validation settings. In order to support the different inference cases of all the MTP problem settings, we define the following four experimental settings (see Fig. 1) under which one can make predictions for new couples $(\mathbf{x}_i, \mathbf{t}_j)$:

- **Setting A**: Both $\mathbf{x}_i$ and $\mathbf{t}_j$ are observed during training.
- **Setting B**: All targets $\mathbf{t}_j$ are observed during training and the goal is to make predictions for unseen instances $\mathbf{x}_i$.
- **Setting C**: All instances $\mathbf{x}_i$ are observed during training and the goal is to make predictions for unseen targets $\mathbf{t}_j$.
- **Setting D**: Neither $\mathbf{x}_i$ nor $\mathbf{t}_j$ is observed during training.

Problems like multi-label classification, multivariate regression, and multi-task learning are mainly associated with Setting B, as they are inductive w.r.t. instances and transductive w.r.t. the targets. This means that during testing, the model is expected to encounter previously-unseen instances, while all targets will be known beforehand. This characteristic informs us about the user's intentions and is determined by two of the questions in our questionnaire, specifically **Q1** and **Q2**. But, despite the intentions of the user, his/her answers to questions **Q3** and **Q4** are what determines the feasibility of generalization. A basic rule one can use is that if we want to achieve generalization to new instances (targets), appropriate side information should be available for those instances (targets). This is why Setting A is usually associated with matrix completion, as in this problem setting no side information is available for instances or targets and thus no generalization is possible for either of them. Finally, Setting D is considered the most challenging of the settings, as

Table 1 A summary of specific combinations of answers to our purpose-build questionnaire for which an MTP problem setting can be assigned

Q1	Q2	Q3	Q4	Q5	Q6	MTP method
Yes	No	Yes	No	Yes	Binary	Multi-label classification (Yan et al. 2019; Hua et al. 2020; Baumel et al. 2018)
Yes	No	Yes	No	Yes	Real-valued	Multivariate regression (Raj et al. 2020; Qi and Roe 2016)
Yes	No	Yes	No	No	–	Multi-task learning
Yes	No	Yes	Yes (hierarchy)	Yes	Binary	Hierarchical multi-label classification (Wehrmann et al. 2018; Zhang et al. 2017; Chen et al. 2019a)
Yes	No	Yes	Yes	No	–	Dyadic prediction (Jin et al. 2017; Liu et al. 2017c)
Yes	Yes	Yes	Yes	No	–	Zero-shot learning (Romera-Paredes and Torr 2015; Mishra et al. 2018)
No	No	No	No	No	–	Matrix completion (Zhao et al. 2016)
No	No	Yes	Yes	No	–	Hybrid Matrix completion (Deldjoo et al. 2019)
Yes	Yes	Yes	Yes	No	–	Cold-start Collaborative filtering (Wei et al. 2016)
Yes	No	Yes	No	Yes	Nominal/categorical	Multi-dimensional classification (Jia and Zhang 2020; Bielza et al. 2011; Shatkay et al. 2008)
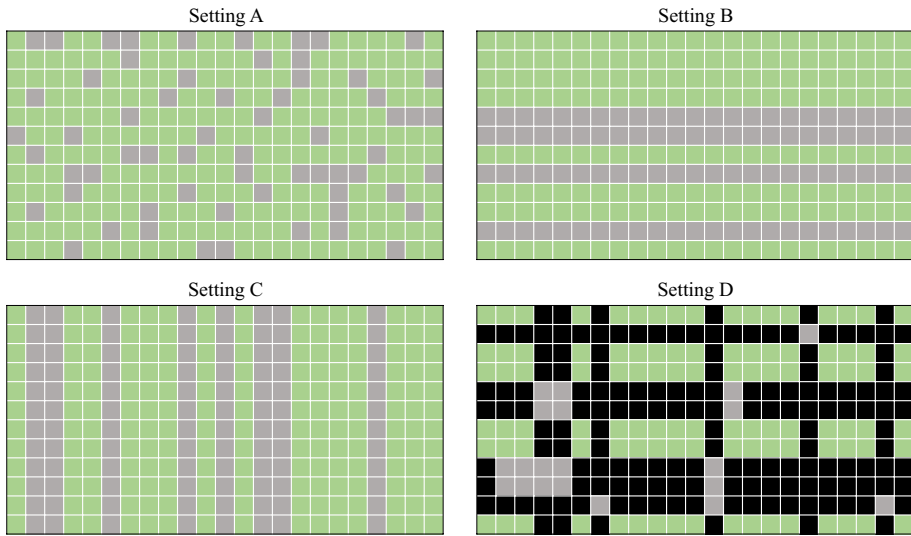
Fig. 1 The four validation settings supported by the DeepMTP framework visualized for the same interaction matrix. Each row corresponds to a different instance $\mathbf{x}_i$ and each column to a different target $\mathbf{t}_j$. Cells coloured in green correspond to known values $y_{i,j}$ present in the training set. The grey cells represent the missing values or values belonging to the test set. Every black cell in setting D is purposely excluded from both train and test sets. In Setting A the test set is formed by randomly sampling couples $(\mathbf{x}_i, \mathbf{t}_j)$ from the interaction matrix. In Setting B the test set is comprised of entire rows of the interaction matrix, which translates to all possible couples $(\mathbf{x}_i, \mathbf{t}_j)$ for specific targets $\mathbf{t}_j$. Setting C can be seen as the converse of Setting B, as in this case the test set includes all possible couples $(\mathbf{x}_i, \mathbf{t}_j)$ for specific instances. Finally, in Setting D the test set has to contain couples $(\mathbf{x}_i, \mathbf{t}_j)$ of which both instance and target are excluded from the train set

the goal is to make predictions for pairs of unseen instances and targets. In the literature on multi-task and transfer learning, this setting is known as zero-shot learning.
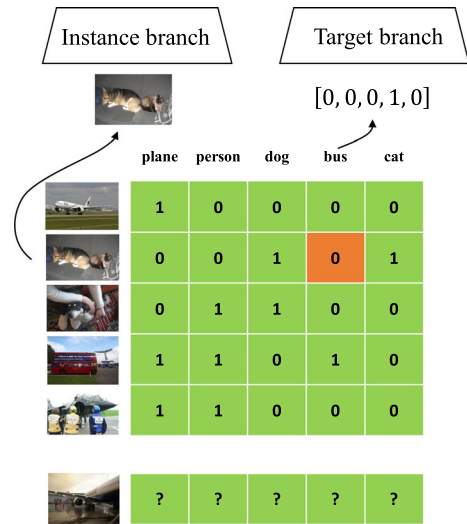
3 From real-world problems to MTP problem settings

This section details real-world examples that map to four of the most popular MTP problem settings (multi-label classification, dyadic prediction, matrix completion, and multi-task learning). For each of these examples, we explain how specific characteristics of the datasets and common requests from the end-user provide answers to the queries of our purpose-built questionnaire. Readers already familiar with the various MTP settings might consider to skip this section.

3.1 Multi-label classification

A typical example of a multi-label classification problem is that of image tagging shown in Fig. 2. A user of our framework who wishes to solve a similar problem will have to possess a dataset that contains images (instances) and their known annotations from a set of possible tags (targets). His/her goal will be to annotate new images (**Q1=yes**) with the tags that were available in the training set (**Q2=no, Setting B**). The pixel values of the images

Fig. 2 Example of a multi-label classification problem from the area of image tagging. Rows represent the different images and columns represent all the possible labels that can be associated with an image. The interaction matrix is fully observed with binary values (1 if the label is associated with the image, 0 otherwise). Side information for the instances corresponds to the raw pixel values of the image. Additionally, because tags are usually not described by side information, we automatically generate one-hot encoded vectors. After the model is trained, testing involves predicting for a new image whether each of the tags should be associated with it

Instance branch		Target branch		
		[0, 0, 0, 1, 0]		
	plane	person	dog	bus	cat
	1	0	0	0	0
	0	0	1	0	1
	0	1	1	0	0
	1	1	0	1	0
	1	1	0	0	0
	?	?	?	?	?

constitute the side information for the instances (**Q3=yes**) in our DeepMTP framework. At the same time, because the tags usually do not contain any kind of side information (**Q4=no**), we have to produce one-hot encoded vectors in order to feed the corresponding branch of our neural network. The one-hot encoded vectors have the same length as the total number of targets and all the positions except one are filled with zeros. The position that maps to the unique id of a target is filled with a one. The problem is considered as a classification problem because the tags have a binary relationship with a given image; they can either be associated with that image or not (**Q6=binary**). The combination of all those characteristics and the specific answers they correspond to in our questionnaire leads us to the identification of the task as a multi-label classification problem.

It is important to point out that there are also instances of similar image tagging tasks that also offer a tag hierarchy (**Q4=yes=hierarchy**). In such an example, all the other characteristics are the same as what we presented in the paragraph above. Instead of creating a standard one-hot encoded vector, we use the position of each target inside the given hierarchy to create a new vector that is passed to the corresponding branch. The availability of additional side information for the targets sets this task apart as a hierarchical multi-label classification problem. Information in the form of a hierarchy might also appear in other MTP problem settings such as multivariate regression, but we are not aware of any publicly-available datasets or even research areas with appropriate naming.

3.2 Dyadic prediction

Dyadic prediction problems can be found in the field of drug discovery and, more specifically, in the task of predicting the interaction between chemical compounds and proteins (shown in Fig. 3 and known as drug-target interaction prediction or DTI). A typical dataset in this area contains interaction information in the form of real-valued affinity scores (**Q6=real-valued**) between proteins (instances) and chemical compounds (targets). Usually, both of these types of molecules are described by vector representations (**Q3=yes, Q4=yes**) that can be found in popular databases (PubChem Kim et al. (2021), DrugBank Wishart et al. (2006), ChEMBL Gaulton et al. (2012)). In a real-world environment, a user,

Fig. 3 Example of a dyadic prediction problem from the field of drug-target interaction prediction. Rows represent the different proteins and columns represent the chemical compound library that a pharmaceutical company might have. The interaction matrix is fully observed and every real value corresponds to the binding affinity of the drug-protein pair
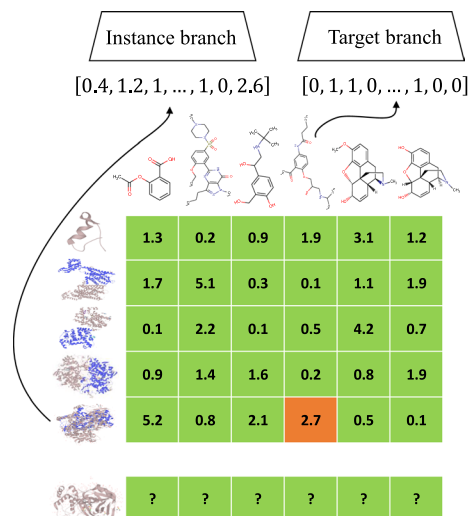

usually a scientist working on a particular disease, identifies a new protein as a potential target for that disease. His/her goal is to check the degree of interaction of that new protein (**Q1=yes**) with every chemical compound in the aforementioned chemical library (**Q2=no, Setting B**). The combination of the dataset's properties with the needs of the user leads us to characterize the task as a dyadic prediction problem. It is useful to note that we could easily interchange the role of the proteins and the chemical compounds in our framework, while still considering it a dyadic prediction problem.

3.3 Matrix completion

The wide-spread acceptance of e-commerce by companies and customers alike has already generated a significant amount of data that can be used to individualize product recommendations. This has resulted in rapid advancements in the area of recommender systems, which aim to predict the users' interests and recommend items that are likely to be interesting to them. A typical dataset from this area of matrix completion contains some kind of interaction between users (instances) and items (targets). This interaction can be expressed in terms of a binary value (someone bought a product or not) (**Q6=binary**) or a real value (someone gave a rating to a movie) (**Q6=ordinal**).

Another characteristic of this type of dataset is that there is information for only a subset of all the possible pairs (**Q5=no**). For example, it is only natural that a user cannot rate every movie in a library of thousands. The objective of this task is to make recommendations by completing the interaction matrix that the already seen users (**Q1=no**) and items (**Q2=no**) create, while no side information is known for either of them (Fig. 4). When side information is available (user's profile page and/or general information about the movie-series) it can be used to potentially improve the performance in the completion task (Hybrid Matrix Completion).

An extension of this formulation leads to the Cold-start Collaborative filtering problem that can be seen as the result of the continuously-evolving nature of the user-base of many companies. This necessitates the prediction of interactions for new users that were not present in the dataset that the original model was trained on (**Q1=yes**). By reversing the role
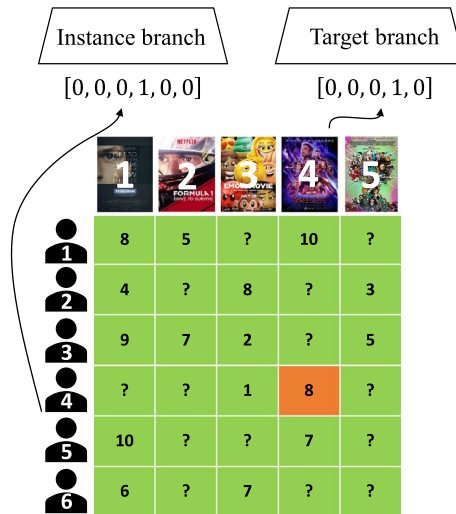
Fig. 4 Example of a matrix completion problem from the broader area of collaborative filtering. Rows represent the different users of a streaming company and columns represent digital content that belongs to its library. Values inside the interaction matrix represent the ratings that the users have given to the content. In this problem, it is expected that the interaction matrix has mainly missing values as it is impossible for a user to rate every movie and series of the company's library. In the standard matrix completion setting, users and movies are not described by side information, so our framework uses their unique id to construct one-hot encoded vectors. The absence of side information also limits the model to only predict ratings for pairs with known users and movies. When side information is actually available, it is possible to extend prediction to pairs with previously unknown users or movies

of instances and targets, the same argument could be made for new items (**Q2=yes**) that are added to the database of a company. For example, when a new movie is available on a platform, the objective could be to first predict the expected rating of each user, and then suggest it to the ones that would give high ratings. Such a generalization is only possible if the appropriate side information becomes available (**Q1=yes** and **Q3=yes**; or **Q2=yes** and **Q4=yes**).

3.4 Multi-task learning

In contrast to well-defined MTP problem settings like multi-label classification and multivariate regression, multi-task learning contains multiple sub-categories of problems. It thus is more challenging to give a concise definition. A large proportion of work published in this area actually works on problems containing different types of variables for each task (heterogeneous tasks). The pairwise manner in which DeepMTP performs training combined with the use of a single type of loss function during the entire training phase makes the heterogeneous setting incompatible. For example, if our architecture was trained for a multi-task learning problem with two heterogeneous tasks (one binary and one real-valued), we would need two different loss funtions (BCE for the values in the binary task and MSE for the real values in the regression task). This is currently not possible; in the next section, we will explain that our neural network architecture optimizes only one loss per problem.

A task that suits this setting's characteristics can be found in the area of crowdsourced annotation (Liu et al. 2018). The quality of training data has been a major limiting factor for the improvement of performance in supervised and semi-supervised tasks. The increasing size of datasets, combined with the high cost of annotating, has led many researchers and companies to crowdsourcing. A user that has a dataset that needs to be annotated can use a crowdsourcing service in order to obtain labels. The resulting dataset he/she will get back could be arranged in an interaction matrix, where the instances map to the original samples of the dataset and the targets map to the annotators. Figure 5 shows a similar example where the instances correspond to documents for which we have the raw text (**Q3=yes**), and the targets correspond to users that are identified by their id (**Q4=no**). Depending on the number of possible labels that a user can assign to a document, the interaction matrix can have binary (Fig. 5, left) (**Q6=binary**) or nominal (Fig. 5, right) (**Q6=nominal**) values. Such a dataset with binary annotations leads to a binary multi-task learning problem, while multi-class annotations lead to a multi-class multi-task learning problem.

A binary multi-task version can also be created if we replace every user's original annotation with a binary value that expresses whether the annotation is correct. Because the size of datasets that need to be annotated is usually close to hundreds of thousands or even millions, it is not feasible for every user to annotate every sample (**Q5=no**). Finally, during inference, the goal could be to predict how every known user (**Q2=no**) would annotate a new, previously unseen document or even if these annotations would be correct.

4 A two-branch neural network architecture for MTP

The baseline architecture of our framework was first popularized by the neural collaborative filtering (NCF) framework (He et al. 2017) in the field of recommender systems. The architecture successfully approximated standard matrix factorization techniques and
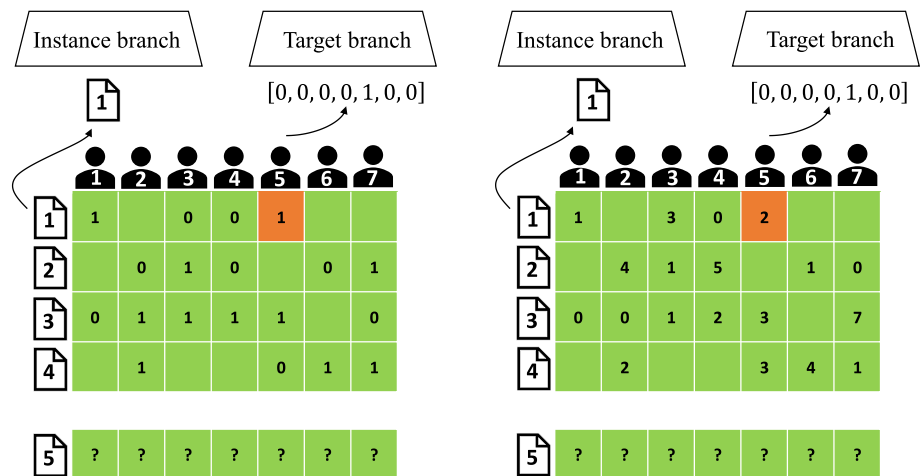

Fig. 5 Examples of multi-task learning problems from the field of crowdsourced annotation. The figure on the left maps to a binary multi-task learning problem because the values in the interaction matrix are binary. The figure on the right represents a multi-class multi-task learning problem, as the values in the interaction matrix are nominal. All the other characteristics in both figures are identical

showed state-of-the-art performance on benchmark datasets. In this work, we show how we can enhance the basic principles of the NCF framework in order to build a generalized framework that achieves a competitive performance in all the settings that fall under the umbrella of MTP.

In the proposed architecture shown in Fig. 6, the network uses two branches to encode the inputs. More specifically, the bottom input layer of each branch is comprised of two feature vectors $\mathbf{x}_i$ and $\mathbf{t}_j$, which describe the instance and target of a sample in an MTP problem. Both vectors can be customized to support a range of different MTP formulations. For example, in a typical multi-label classification problem, a one-hot encoded vector will be generated to represent a specific target and used as input to the corresponding branch. Using the same principles, in a typical matrix completion problem, we will have to generate one-hot encoded vectors for both instances and targets using their unique ids, very similar to what NCF does.

Above the input layer, we extend the NCF framework by using different types of layers or even entire sub-architectures to better encode the different kinds of inputs the framework
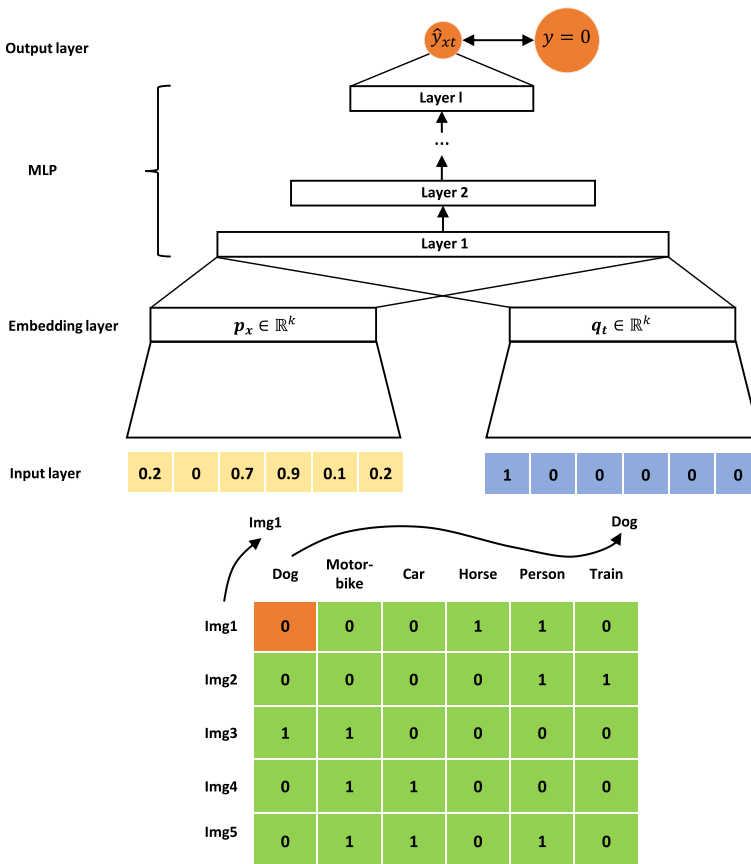
Fig. 6 Detailed view of the two-branch neural network. The specific example shows an image tagging problem where the one-hot encoded vectors for the pixel values and the tag are fed into the corresponding branches, transformed into embedding vectors, and then passed to an MLP that outputs the predicted interaction score

may encounter. In cases where no side information is provided (for example, the labels in a multi-label classification problem), we use a single fully-connected layer to project the sparse one-hot encoded input vector to a dense embedding. Otherwise, when explicit side information is available, we have multiple options, depending on the type of input, from several fully-connected layers (tabular health record data, Fig. 7, left) to more specialized architectures based on convolutional neural networks (Fig. 7, right) or graph neural networks (hierarchies). The goal of the embedding layer in both cases is to project the instances and targets to a lower-dimensional latent space, similarly to what is done with the users and items in the product recommendation problem in NCF (He et al. 2017).

The instance embedding $\mathbf{p_x}$ and target embedding $\mathbf{q_t}$ are then concatenated and passed through a multi-layer neural network architecture that maps the embeddings to the predicted target value in the following way:

$$
\begin{aligned}
\mathbf{z}_1 = \phi_1(\mathbf{p_x}, \mathbf{q_t}) &= \begin{bmatrix} \mathbf{p_x} \\ \mathbf{q_t} \end{bmatrix}, \\
\phi_2(\mathbf{z}_1) &= \alpha_2(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2), \\
&\ \ \ldots\ldots \\
\phi_L(\mathbf{z}_{L-1}) &= \alpha_L(\mathbf{W}_L^T \mathbf{z}_{L-1} + \mathbf{b}_L), \\
\hat{y}_{\mathbf{xt}} &= \sigma(\mathbf{h}^T \phi_L(\mathbf{z}_{L-1})),
\end{aligned} \tag{1}
$$

where $\mathbf{W}$, $\mathbf{b}$ and α correspond to the weight matrix, bias vector and activation function of the final multi-layer perceptron (MLP) layer. We mainly use the leaky rectified linear unit (Leaky ReLU) as activation function in our framework, but because we also perform experiments with custom architectures from third parties instead of the branches, other activation functions may also be utilized (for example, standard ReLU in Resnet He et al. 2016).

This MLP architecture is able to model more complex, non-linear instance-target relationships compared to a simpler dot product. Even though this idea was popularized by the NCF framework and widely adopted by the CF community, there has been recent work proposing that the dot product may be highly competitive and cheaper to train (Rendle
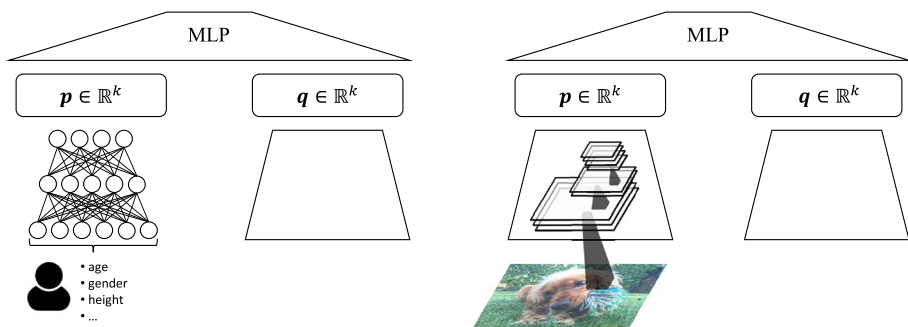
Fig. 7 Examples of different architectures that can be used in the branches of the multi-branch neural network. In the left figure, we use a conventional fully connected neural network because the input consists of tabular user-related features, whereas in the right figure, we use a convolutional architecture because the input is in the form of images. Both versions of our dual branch architecture utilize a final multi-layer perceptron (MLP) that takes as input the vector obtained by concatenating the instance embedding vector $\mathbf{p_x}$ and the target embedding vector $\mathbf{q_t}$

et al. 2020; Dacrema et al. 2021). Regardless, we decided that all the experiments shown below should use an MLP and that we will investigate whether the dot product can be a viable alternative for the MTP settings in future work.

The final output layer consists of a single node that outputs the predicted score $\hat{y}_{\mathbf{xt}}$. In the classification-related MTP settings a sigmoid function is used before the output in order to restrict it to [0, 1]. We facilitate training using different loss functions to accommodate the different categories of MTP problem settings. In classification problems, training is achieved using the binary cross-entropy loss function:

$$L_{\text{BCE}} = - \sum_{(\mathbf{x},\mathbf{t},y)\in\mathcal{D}} y \log \hat{y}_{\mathbf{xt}} + (1 - y) \log (1 - \hat{y}_{\mathbf{xt}}) . \qquad (2)$$

On the other hand, in problems that fall into the regression category, we use the squared error loss:

$$L_{\text{MSE}} = \sum_{(\mathbf{x},\mathbf{t},y)\in\mathcal{D}} (y - \hat{y}_{\mathbf{xt}})^2 . \qquad (3)$$

In both loss functions, $\mathcal{D}$ denotes the set of known interactions in the training set.

In order to make it more accessible to the reader how training and inference work in our architecture, we make a comparison with a standard neural network in the popular multi-label classification case shown in Fig. 6. The basic neural network will have as many input nodes as instance features and as many output nodes as there are labels (six in the example). This means that for the example in Fig. 6, the neural network will use the pixel values of an image as input and then output the prediction for every label simultaneously. This procedure is followed during training as well as inference. In our architecture, training and inference are performed in a pairwise manner. Instead of working with all the labels of an image simultaneously, we process each instance-target pair separately. Thus, for the same example we detailed earlier, our network will have to input the same image six times to the instance branch and modify the one-hot encoded vector that is passed to the target branch.

It is also important to point out that there are cases in which additional side information is available. These features are usually available for every couple $(\mathbf{x}_i, \mathbf{t}_j)$ in the dataset and have been coined dyadic features in the literature (Van Peer et al. 2017). Such information requires an extension of our two-branch architecture by a third branch that allows to encode those dyadic features (Fig. 8, right). Similar architectures have been successfully deployed in tensor factorization applications (Wu et al. 2018; Schreiber et al. 2020). In this setting, training and inference remain largerly unchanged, the only difference being the concatenation of three embedding vectors $\mathbf{p}_\mathbf{x}$, $\mathbf{q}_\mathbf{t}$ and $\mathbf{r}_\mathbf{d}$ instead of just two.

Finally, our neural network architecture, combined with the pairwise manner in which we train our models, allows to make predictions for all four validation settings shown in
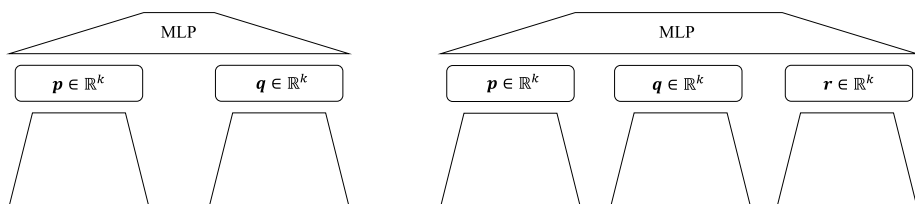

Fig. 8 General two-branch architecture (left) and tri-branch architecture (right)

Fig. 1 (Settings A, B, C and D) without having to make modifications in the core training and inference steps. The only stages in the pipeline that need to be adapted are the preparation of the dataset splitting as well as the computation of the performance metrics. In the experiments presented in Sect. 6, we only report results for Settings A and B, as they are the most frequently encountered ones. In future work, we intend to also report the performance for the two other settings and discuss the differences between them.

5 Related work

This section's goal is to discuss related work. The literature on multi-target prediction is vast, so we will focus on deep learning approaches for multi-target prediction. First, we review two-branch neural network architectures that have been introduced for specific problem settings (some of thse settings fall under the MTP umbrella). Those architectures are often very similar to the architecture we propose. Second, we review other deep learning methods that can be used for multi-target prediction, i.e. architectures that are not based on two branches. Third, we briefly discuss some well-known MTP methods that are not at all based on neural networks.

Two-branch neural network architectures have been developed for distance metric learning, similarity learning and object matching problems. In these application domains, such architectures are often referred to as Siamese neural networks (Bromley et al. 1993). The architecture typically consists of two identical branches, which are both capable of learning the hidden representation of an input vector. The two outputs are then compared, usually through cosine similarity, and the output of such a network can be thought of as the semantic similarity between the two embedding vectors. Siamese neural networks have found extensive use in video analysis (Ryoo et al. 2018; Liu et al. 2017b), but also in audio processing (Pitt et al. 2005; Chen and Salman 2011) and natural language processing (Yih et al. 2011; Marelli et al. 2014; Das et al. 2016). For a more extensive review of the application of Siamese networks, we refer to Chicco (2021).

Two-branch neural networks can also be used to learn the similarity between two objects of a different type. In this setting, the two branches will have a different architecture, similar to our framework. In computer vision one can find several papers that adopt such an idea for different applications, without a focus on developing general-purpose tools. Convolutions are used in the branch that encodes images, while other layer types are considered in the second branch. As representative examples, let us discuss three papers a bit more in detail. Wang et al. (2018) investigate two-branch neural networks to learn the similarity between image and text modalities for the purpose of phrase localization and bi-directional image-sentence retrieval. Shao and Qian (2019) consider a two-branch convolutional neural network to classify facial expressions. The first branch takes as input the raw image and extracts global features, while the second uses local binary pattern features to extract local texture features. As a third example, Pan et al. (2018) introduce DualCNN for various low-level vision problems like super-resolution, noise/artifact removal, image deraining, and dehazing. Their architecture consists of two branches, one shallow sub-network to estimate the structures of the input image and one deep sub-network to estimate the details.

In recent years two-branch neural networks have also been introduced in recommender systems. In fact, the neural collaborative filtering framework of He et al. (2017), which has been explained in Sect. 4, has become one of the most popular neural-network-based matrix factorization methods. One of the methods proposed in He et al. (2017), called generalized

matrix factorization, computes the dot product between the two branches, but this is only possible when the learned embeddings of the two branches have the same dimensionality. Moreover, the dot-product is not parameterized by any additional (learnable) parameters, which might hamper the predictive performance. That is why they also suggest a modification that is used in our work in which the learned embeddings are concatenated to a single vector that serves as input for another fully-connected feed-forward neural network. As another alternative, He et al. (2018) use an outer product to explicitly model the pairwise correlations between the dimensions of the embedding space. The outer product creates a two-dimensional interaction map that is then processed by a convolutional neural network to learn high-order correlations among the embedding dimensions effectively.

A natural extension of the use of two branches for matrix factorization is the inclusion of a third branch that can encode a third dimension and thus be used for tensor factorization. Wu et al. (2019) introduce a neural-network-based tensor factorization model that contains a third (LSTM-based) branch to characterize the multi-dimensional temporal interactions for relational data. For some applications, we believe that it is also relevant to include a third branch in our MTP framework, and this is something we will experiment with in the future.

In multi-label classification, several deep learning methods that do not consider two branches have been presented. Gong et al. (2013) used a convolutional architecture, similar to what we did for any task that involved images and experimented using different ranking-based loss functions. He demonstrated that the weighted approximated ranking loss, which specifically optimizes the top-‖ accuracy (not possible with the current version of our work), works well for multi-label annotation problems. Nam et al. (2017) propose a sequence-to-sequence recurrent neural network as an alternative to the well-known classifier chains method. Similar to other chaining methods, this neural network is mainly useful for optimizing the subset zero-one loss (not considered in this paper). In the area of multi-label image classification, Wang et al. (2016) combine deep convolutional and recurrent neural networks in a framework that is able to learn a joint image-label embedding that exploits label dependencies. Because this approach uses LSTMs, a predefined label ordering is required during training, something that is usually not available. For that reason, Chen et al. (2018) investigate the effectiveness of a deep learning model that combines a visual attention model with an LSTM and thus does not require any predetermined label ordering. Huynh and Elhamifar (2020) consider a shared multi-attention mechanism that predicts all seen and unseen labels in an image, something that other attention-based approaches are unable to do. Finally, custom architectures have also been proposed for cases in which the number of labels becomes very large (extreme multi-label classification). Liu et al. (2017a) used deep convolutional neural networks for multi-label text classification and showed a competitive performance for datasets with up to 670k labels. In the same area, Zhang et al. (2018) established an explicit label graph to better model the label space of extreme multi-label classification datasets. Our approach is able to scale linearly with the number of labels, but further work will be needed to improve speed and make experimentation with larger datasets feasible.

Lastly, as far as software packages go, we could not find any work that provides methods for more than two MTP problem settings. Tsoumakas et al. (2011) developed an open-source Java library that implements several transformation methods like binary relevance, label powerset, and other multi-label algorithms like multi-label k nearest neighbors, random k-labelsets, the hierarchy of multi-label learners algorithm, and back-propagation multi-label learning. In contrast to the command-line interface of Mulan, MEKA Read et al. (2016) is another popular Java library that provides a

graphical interface and inherits methods implemented in Weka Hall et al. (2009). Another open-source library that was introduced more recently and is written in python is called scikit-multilearn (Szymański and Kajdanowicz 2017). This library can utilize methods from scikit-learn and provides an interface for MEKA, but the set of methods included is limited. Finally, the MLC toolbox (Kimura et al. 2017) offered multi-label classification methods for MATLAB/OCTAVE users.

6 Experimental results on various MTP problems

This section's main goal is to convey that our architecture is flexible enough to train and make predictions with minimal configuration changes for multiple MTP problem settings. We also want to showcase that our approach is quite competitive with methods that are usually purpose-built for only one of the problem settings. Of course, for the same reason, we do not expect and is generally not our goal to outperform all the methods we are comparing with. At the end of this section, we anticipate that our framework will constitute a viable benchmark for future methods that will be developed for any of the MTP problems settings we have explored. The experimental setup as well as the hyperparameter space of the methods we compare with are located in the Appendix.

6.1 Multi-label classification

For the multi-label classifcation problem setting, we selected methods that are available in the scikit-learn (Pedregosa et al. 2011) and scikit-multilearn (Szymański and Kajdanowicz 2017) libraries. More specifically, we compare with a standard neural network in which the number of output nodes is equal to the number of targets, two instances of a binary relevance approach that has a support vector machine (SVM) and a neural network as base classifier, a nearest neighbors method adapted for multi-label classification (MLkNN) (Zhang and Zhou 2007), a multi-output decision tree classifier (DT), and an ensemble of classifier chains (ECC) (Read et al. 2009) that uses an SVM as the base classifier. Experiments were performed using four benchmark datasets from Mulan's GitHub repository (Tsoumakas et al. 2011). Table 2 lists these datasets along with their main statistics. Because there is no target side information in this setting, our framework uses one-hot encoded vectors as inputs for the corresponding branch.

The hyperparameters of the methods we compare with were optimized through a grid search. The performance metric of choice for this problem setting is the widely used Hamming loss, which the majority of methods can explicitly optimize for. The importance of this characteristic was originally explored in Dembczyński et al. (2010) and influenced the selection of the methods we compare with. The results shown in Table 2 illustrate the competitiveness of the DeepMTP framework, as it achieves comparable performance to the other baselines on all four datasets.The experimental section for this MTP problem setting is not as extensive as other papers that are exclusively focused on this area, both in terms of datasets and methods. This is done purposely, as we have to consider many other MTP settings. For extensive comparisons in this area, we refer to work by Madjarov et al. (2012); Tsoumakas and Katakis (2007).

Table 2 The four multi-label classification data sets used in this study and reported Hamming loss of every method for these datasets

	Yeast	Scene	Bibtex	Corel5k
Data properties				
No. of instances	2417	2407	7395	5000
No. of instance features	103	294	1836	499
No. of targets	14	6	159	374
No. of target features	14	6	159	374
Hamming loss				
BR (SVM)	**0.1935**	**0.0733**	**0.0130**	0.0146
BR (MLP)	0.2223	0.0993	0.0142	0.0201
MLP	0.2406	0.0897	0.0198	0.0183
MLkNN	0.202	0.0885	0.0146	0.0162
DT	0.261	0.1301	0.0137	0.0151
ECC (SVM)	0.2841	0.1312	0.0148	0.0152
DeepMTP	0.2309	0.0839	0.0157	**0.0114**

Results shown in bold indicate the best performance for every dataset

6.2 Multivariate regression

Similarly to the previous setting, all the selected methods were obtained from the scikit-learn library (Pedregosa et al. 2011). The methods selected in this setting include a typical multilayer perceptron (MLP), popular single-target approaches [support vector regression (SVR), Kernel ridge regression (KRR), decision tree regression], as well as an ensemble of 50 regressor chain models that use a support vector regressor as base model. We also selected the seven datasets listed in Table 3 from a repository that accompanied (Melki et al. 2017).

Table 3 The seven multivariate regression data sets used in this study and reported aRRMSE of every method for these datasets

	Enb	Jura	Water quality	Oes97	Oes10	Puma8nh	Puma32h
Data properties							
No. of instances	768	359	1060	323	403	8192	8192
No. of instance features	8	11	16	263	298	8	32
No. of targets	2	7	14	16	16	3	6
No. of target features	2	7	14	16	16	3	6
aRRMSE							
SVR/target	0.1161	**0.5747**	0.9493	0.5394	0.3410	0.8818	0.9634
DTR	0.1629	0.7587	1.0310	0.7620	0.5024	0.9590	1.0949
KRR/target	0.1606	0.5785	0.9500	0.5312	0.3468	0.8625	1.0001
MLP	**0.0933**	0.6334	0.9809	0.7885	0.3946	0.8750	1.0008
ECC (SVR)	0.1231	0.5751	0.9472	0.5393	**0.3409**	0.8780	**0.9633**
DeepMTP	0.0954	0.6614	**0.9279**	**0.4843**	0.4292	**0.8509**	1.002

Results shown in bold indicate the best performace for every dataset

The hyperparameters of the methods we compare with were optimized through a grid search. The performance metric used in this setting is the commonly-used average relative root mean square error (aRRMSE). The results shown in Table 3 indicate that our approach is quite competitive, outperforming the other methods on three out of the six available datasets. The DeepMTP framework's performance closely resembles that of the standard neural network and becomes more competitive when the number of training samples increases.

6.3 Hierarchical multi-label classification

For the hierarchical multi-label classification problem problem setting, we selected two image classification datasets that included hierarchical information for the targets, which in this case corresponds to tags that can be associated with an image. More specifically, these datasets are the MSCOCO and the VOC 2007, two really popular benchmarks in the area of multi-label classification. Microsoft COCO (Lin et al. 2014) is a benchmark that contains 82081 images in the training set and 40504 images in the validation set. There are also 80 different labels that can be associated with an image with the actual average being 2.9 labels per image.

The second dataset used in this setting comes from the PASCAL Visual Object Classes Challenge (VOC 2007) (Everingham et al. 2010) and is divided into train, validation and test sets. This benchmark contains 9963 images and 20 different tags that are organized in the hierarchy shown in Fig. 9. In our experiments, the methods were trained on both the train and validation sets and the evaluation was done using the test set.

In terms of the configuration of our framework for this problem, we decided to use a pre-trained version of the ResNet-101 architecture, similar to what is shown in the right of Fig. 6, as well as Fig. 2. For the branch that encodes the targets, we experimented with two different versions. In the first one, we create standard one-hot encoded vectors, similarly to what we do when no side information is available. In the second version, we utilize the available tag relations by constructing sparse vectors that encode the given hierarchy. For example, inspecting the hierarchy for the VOC 2007 dataset in Fig. 9, we count nine categories and 20 final classes-tags. To construct a vector that encodes the hierarchy, we first create a 29-dimensional vector populated by zeros. Each position of the vector maps to a different category or tag. Then, to represent a specific tag we start from the root of the
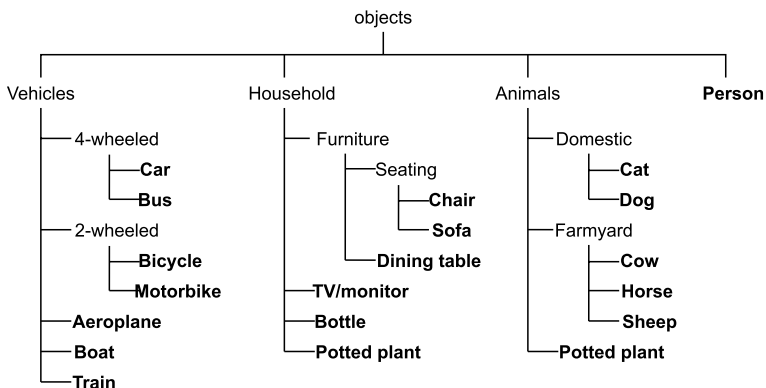

Fig. 9 Hierarchy of the 20 categories present in the VOC 2007 dataset

hierarchy and traverse it until we arrive at that tag. For each node we encounter, we assign a one to the corresponding position in the vector.

In terms of methods, we decided to compare with a graph-convolution-based approach that was proposed in Chen et al. (2019b). In the latter paper, the authors present experiments with the two datasets presented above. Even if theoretically the use of the same train-test split would not necessitate re-running their experiments in order to compare with our framework, we decided to do so using their published implementation. The results on MS-COCO and VOC2007 are shown in Table 4. In terms of metrics, we decided to use the same as in Chen et al. (2019b). More specifically, we computed the macro-wise (macro-P, macro-R, macro-F1) and instance-wise (inst-P, inst-R, inst-F1) versions of recall, precision and F1 score.

From the results presented above, we observe that the DeepMTP framework shows a competitive performance with the ML-GNC method. It is also important to mention that in our experiments ML-GNC achieved a slightly worse performance across all six metrics compared to what is reported in Chen et al. (2019b). The same paper makes comparisons with multiple other methods, for which we could not find any implementation. We hypothesize that this is also the reason why some performance values are missing from their table of results (the source papers report a subset of the six metrics they use, so they could not copy these results). This is the reason why we do not include these methods in the present work. Finally, we report that our experiments with the two different versions of target features did not result in a significant difference in performance. This can be explained by the fact that both of the used hierarchies are quite shallow, and thus do not offer useful information compared to the standard one-hot encoded features. We believe that the ability of our framework to easily include or disregard the

Table 4 Comparison of the ML-GNC and DeepMTP methods in terms of multiple performance metrics on the MS-COCO dataset

	MS-COCO	VOC2007
Instance-precision		
ML-GNC	**0.858**	**0.8725**
DeepMTP	0.8344	0.8612
Instance-recall		
ML-GNC	**0.754**	**0.9107**
DeepMTP	0.6559	0.8035
Instance-F1		
ML-GNC	**0.803**	**0.8912**
DeepMTP	0.7345	0.8313
Macro-precision		
ML-GNC	**0.851**	**0.8602**
DeepMTP	0.7885	0.8364
Macro-recall		
ML-GNC	**0.72**	**0.8947**
DeepMTP	0.6144	0.8004
Macro-F1		
ML-GNC	**0.78**	**0.8771**
DeepMTP	0.6807	0.8137

Results shown in bold indicate the best performace for every metric and dataset

hierarchical information boosts the potential of our framework for the hierarchical classification setting.

6.4 Matrix completion

For this task, we decided to compare with methods that are available in Microsoft's repository of recommender systems (Microsoft 2018). Because we do not support a ranking loss at this stage, we only included methods that optimize for regression. These methods include a matrix factorization approach by alternating least squares (MF-ALS), a neural network approach that is really similar to ours but uses a dot product to combine the instance and target embedding vectors (fastai), the Riemannian Low-rank Matrix Completion (rlrmc), and another neural network approach that trains a wide linear model as well as a deep neural network (wide & deep). In terms of datasets, we decided to use two versions of the widely-used movielens dataset, one with 100 ratings (movielens100k) and one with one million ratings (movielens1M). The dataset contains ratings that users gave to movies, exactly as shown in Fig. 4. The test set is formed by randomly selecting 25% of the known ratings (Setting A). Because we do not have any side information available for either users (instances) or movies (targets), we generate one-hot encoded vectors for both of them. Similarly to what we see with all the other MTP problem settings, the DeepMTP framework is quite competitive. For both versions of the movielens dataset, the performance is quite similar to or even outperforms that of the other methods (Table 5).

	Movielens100k	Movielens1M
RMSE		
MF-ALS	0.9628	**0.8593**
Fastai	0.9431	0.8745
Rlrmc	1.0441	0.8640
Wide and deep	0.9491	0.9574
DeepMTP	**0.9391**	0.8782
MAE		
MF-ALS	0.7488	0.6783
Fastai	0.7443	0.6954
Rlrmc	0.7961	**0.6751**
Wide and deep	0.7561	0.7771
DeepMTP	**0.7421**	0.6904
R2		
MF-ALS	0.2590	**0.4133**
Fastai	0.2853	0.3869
Rlrmc	0.1240	0.4016
Wide and deep	0.2762	0.2651
DeepMTP	**0.2913**	0.3817

Table 5 Comparison of the collaborative filtering approaches in terms of multiple regression performance metrics for movielens100k and movielens1M

Results shown in bold indicate the best performace for every metric and dataset

6.5 Multi-task learning

For the multi-task learning problem setting, we decided to experiment with crowdsourcing datasets, in a very similar setup as described in Fig. 5. This was partly done because contemporary research in multi-task learning works on heterogeneous interaction matrices, something that our framework does not support at this moment.

The datasets we used in this setting were first introduced in Liu et al. (2018). These include two image datasets that are labeled by users. The first one contains 800 images of dogs and 52 annotators that have to label each image with one of four available breeds. The second dataset contains 2000 images of 10 different types of birds that were labeled by 65 annotators. In both datasets, the majority of possible image-user pairs is missing, as it is challenging for a user to annotate thousands of images. To simplify the problem, we used the correct annotations that were supplied for every image to transform the original multiclass, multi-task learning problem into a binary one. A given cell in the final interaction matrix shows whether or not the user labeled an image correctly (Fig. 5, left).

In terms of methods we decided to compare with, we chose two baselines. The first one simply predicts the majority class. For example, if the majority of a user's annotations is correct, we predict that he/she will also label all the test set images correctly. The second approach is the standard single-task approach in which we train a single model for every task separately. Because the side information of the instances corresponds to raw images, we chose the VGG architecture instead of the corresponding branch in our two-branch neural network. More specifically, we used a pre-trained version of the VGG-11 architecture that had every layer's weights, except for the last one, freezed. This was intentionally done to improve running time and also because both datasets did not have enough instances to train such a massive architecture.

In terms of results (see Table 6), it is clear that the datasets used are not large enough to train the neural networks. In terms of accuracy, the majority voting approach is competitive as the test sets in most cases were comprised of only a few samples. The single-target Resnet approach was unable to properly train and was only predicting the majority class. In terms of AUROC and AUPR, and for both datasets, our approach clearly outperforms the other two methods.

6.6 Dyadic prediction

For the dyadic prediction problem setting, we chose to compare with a network inference approach that uses an ensemble of bi-clustering trees (eBICT) Pliakos and Vens (2019) on datasets that are used in that paper. Although the implementation of the eBICT method is not available online, it was kindly provided to us by the authors upon simple request. The four datasets (see Table 7) that we include in our work are heterogeneous interaction networks that are publicly available and commonly used in the field of bioinformatics. For each dataset, the interaction matrix is populated by binary values and side information is available for both instances and targets. Two of the datasets correspond to drug-protein interaction networks and were originally introduced, together with two additional datasets, as a gold standard in the area of DTI prediction. Side information for the drugs amounts to vectors that code for the similarity of their chemical structure, while side information for the proteins comes in terms of similarities based on the alignments of their sequences. The original four datasets were differentiated by the category of the target protein they include:

Table 6 Reported accuracy, AUROC, AUPR of every method on the two multi-task datasets

	Dogs	Birds
Data properties		
No. of instances	800	2000
No. of instance features	3*224*224	3*224*224
No. of targets	52	65
No. of target features	52	65
Macro-accuracy		
Majority voting	**0.7461**	0.6809
Single-target Resnet	0.7043	0.6588
DeepMTP	0.6621	**0.7027**
AUROC		
Majority voting	0.5	0.5
Single-target Resnet	0.5123	0.5105
DeepMTP	**0.611**	**0.6637**
AUPR		
Majority voting	0.7406	0.6192
Single-target Resnet	0.7421	0.6585
DeepMTP	**0.8071**	**0.7555**

Results shown in bold indicate the best performance for every metric and dataset

Table 7 Reported micro-AUROC and micro-AUPR of every method on the four dyadic prediction datasets

	DPI-E	DPI-IC	SRN	ERN
Data properties				
No. of instances	664	204	1821	1164
No. of instance features	664	204	9884	445
No. of targets	445	210	113	154
No. of target features	445	210	1685	445
Micro-AUROC				
eBICT	0.8053	**0.8338**	**0.8169**	0.8536
DeepMTP	**0.8571**	0.8312	0.8166	**0.8874**
Micro-AUPR				
eBICT	**0.555**	0.4248	**0.1686**	**0.4450**
DeepMTP	0.45	**0.4289**	0.162	0.4295

Results shown in bold indicate the best performace for every metric and dataset

nuclear receptors (NR), G-protein-coupled receptors (GR), ion channels (IC), and enzymes (E). In this work, we excluded two of the datasets (NR and GR) because of their very small size, both in terms of number of instances as well as in terms of number of targets.

The remaining two datasets that we used in our work corresponds to regulatory networks for two different micro-organisms. The first dataset concerns an *E. coli* regulatory network (ERN) that contains pairs of transcription factors and genes of the *E. coli* bacterium. The second dataset represents a similar network but with genes from the

Saccharomyces cerevisiae yeast. Here, the side information for both instances and targets consists of expression values.

In terms of performance metrics, we follow what was proposed in Pliakos and Vens (2019). These include the micro-average versions of the area under the precision recall curve (AUPR), as well as the area under the receiver operating characteristic curve (AUROC). Concerning the hyperparameters for the eBICT method, we used the defaults that were proposed in the corresponding paper. The results shown in Table 7 demonstrate the competitiveness of our approach. In terms of AUROC, we outperform the eBICT method on two out of the four datasets and show a similar performance on the remaining two. In terms of AUPR, we manage to outperform the eBICT method on only one dataset, but we remain competitive on the other three. At this point, it is important to state that we only report results for only one of the four validation settings (Setting B) even though in Pliakos and Vens (2019) the authors also experiment with Settings C and D. We argue that this is similar to what can happen in real-world situations, as a user can choose only one type of generalization despite more options being available. In future work, we expect to also compare our performance for the other three settings (A, C, and D), similar to what the eBICT paper presents.

7 Conclusions and future work

In this paper, we proposed a new framework that aims to make all the problem settings that fall under the umbrella of MTP more accessible to the end-user. In order to do so, we introduced a novel, purpose-built questionnaire that distils our understanding of the commonalities and differences that the MTP problem settings display. We also showed examples of how the characteristics of specific real-world problem settings and datasets lead to specific combinations of answers to our questionnaire and ultimately to a specific MTP problem setting being identified.

We then explained how we use the selected MTP setting information to configure a flexible multi-branch neural network. We also showcased all the different modifications we can perform in the network's architecture, starting from how we handle different types of input data to what losses we can use depending on the different types of output values each problem setting offers. Finally, we provided extensive experimental results for five popular MTP problem settings, covering 21 different datasets and 19 different methods. From those results, we are able to show that our architecture can be quite competitive in all five MTP problem settings with minimal modifications, while facing different types of input and output data, different dimensionalities of input features, and different validation strategies. To conclude, we believe that this architecture can be used as a reliable benchmark in future work related to all MTP problem settings.

In terms of limitations, we would like to point out some variations of MTP problem settings that are not recommended for our framework. Our multi-task experiments included datasets with binary values for all the included targets (binary multi-task learning). Datasets with multiple classes (multi-class multi-task learning) could be tackled by replacing the single output node with a number of nodes that is equal to the number of classes. MTP problem settings like multi-dimensional classification could be tackled using the same configuration for the output layer. Also, similarly to the work by Jia and Zhang (2020), comparisons can be made using modified multivariate regression datasets and baseline methods that are used in multi-label classification (Binary Relevance, Classifier chains, Label

Powerset). Datasets that combine heterogeneous targets (for example, binary, multi-class, and real-valued simultaneously) are not suitable for our architecture as the use of a single loss function limits us to multi-task learning problems with homogeneous targets.

Structured output and multi-class prediction problems are settings that many may consider in the multi-target prediction framework. Multi-class problems could be included using the 1-versus-rest decomposition reduction technique. Following this approach, predicting an instance's output boils down to a set of binary prediction tasks even though we remain interested in a single prediction, not multiple ones. Similarly to Waegeman et al. (2019) we argue that for structured output prediction problems, the target space is often infinitely large, and the structure of the target space needs to be exploited for computational reasons during training and inference. Our framework cannot be used for structured output prediction problems where the target space cannot be enumerated (because every potential output will represent a column in the matrix representation used). As a result, we do not recommend to use our framework for problems of that kind.

Despite the extent of this work, there are still many directions we intend to explore in the future. The immediate next step will be to automate the process of hyperparameter tuning in our model. Our architecture displays unusual characteristics, like branches that can have different dimensionalities and types of sub-architectures. For that reason, the process of finding the optimal architecture using a simple technique like grid search or random search becomes practically infeasible. Another direction we could explore is related to the performance difference that is expected when validating in the four settings we discussed in Sect. 2.2.

Furthermore, even though in Sect. 4 we describe both a two-branch and a tri-branch architecture, we only report experimental results using two branches. This interesting, but at the same time quite underdeveloped area of dyadic information could be an option for our future work. The collection of MTP datasets that also contain dyadic information, combined with benchmark results produced by our DeepMTP framework, would give the necessary boost to other researchers to engage in this task.

Finally, the current version of our framework optimizes specific versions of loss functions that are cell-decomposable (Hamming loss). However, in our results section, we also compared with other methods that do not optimize of the same loss as DeepMTP. An attractive next step for our work would also be to extend the range of loss functions that our DeepMTP framework allows to optimize.

Appendix A

The standard MTP problem settings include multi-label classification, multivariate regression and multi-task learning and are formally defined below.

Definition 3 The **multi-label classification** setting is an instance of the MTP framework with the following additional properties:

(P4) All targets are observed during training ($|\mathcal{T}| = m$).
(P5) No side information is available for targets, thus we identify them with natural numbers ($\mathbf{t}_j = j$).
(P6) The score matrix $\mathbf{Y}$ is fully observed.

(P7)   The score set is $\mathcal{Y} = \{0, 1\}$.

Definition 4 The **multivariate regression** setting is an instance of the MTP framework with the following additional properties:

(P4)   All targets are observed during training ($|\mathcal{T}| = m$).
(P5)   No side information is available for targets, thus we identify them with natural numbers ($\mathbf{t}_j = j$).
(P6)   The score matrix $\mathbf{Y}$ is fully observed.
(P7a) The score set is $\mathcal{Y} = \mathbb{R}$.

Definition 5 The **multi-task learning** setting is an instance of the MTP framework with the following additional properties:

(P4)   All targets are observed during training ($|\mathcal{T}| = m$).
(P5)   No side information is available for targets, thus we identify them with natural numbers ($\mathbf{t}_j = j$).
(P6a) The score matrix $\mathbf{Y}$ has missing values.
(P7a) The score set is homogeneous across the columns of $\mathbf{Y}$.

From the properties of the three definitions presented above, we see that multi-task learning can be interpreted as a generalization of multi-label classification and multivariate regression, the main difference being the missing values in the score matrix $\mathbf{Y}$. A common characteristic of the three settings presented above is that they do not use side information for the targets. The utilization of such information leads to the establishment of extensions for these three standard settings with new titles and separate research areas.

The following definitions correspond to MTP problem settings that are able to utilize side information about the target space.

Definition 6 The **hierarchical multi-label classification** setting is an instance of the MTP framework that shares all the properties of multi-label classification with the following updates:

(P5*) Side information is available for the targets ($\mathcal{T} = \{\mathbf{t}_1, \ldots, \mathbf{t}_m\}$) in the form of target relations (usually hierarchies).

Definition 7 The **dyadic prediction** setting is an instance of the MTP framework that shares all the properties of multi-task learning with the following updates:

(P5*) Side information is available for the targets ($\mathcal{T} = \{\mathbf{t}_1, \ldots, \mathbf{t}_m\}$) in the form of a structured representation.

Definition 8 The **zero-shot learning** setting is an instance of the MTP framework that shares all the properties of dyadic prediction setting with the following updates:

(P4*) Novel targets are expected at prediction time ($|\mathcal{T}| = m^* > m$).

Properties P5 and P6 introduce the notion of inductiveness and transductiveness for instances and targets. For example, the three standard MTP problem settings are inductive w.r.t. instances and transductive w.r.t. targets, as predictions need to be produced for novel instances but not for novel targets. The utilization of side information for the targets is what gives the extended MTP problem settings the ability to generate predictions for novel targets. The following definitions showcase MTP problem settings that arise from the availability of side information for both instances and targets, as well as the intent to generalize to novel instances and/or targets.

Definition 9 The **matrix completion** setting is an instance of the MTP framework that shares properties P4, P5 and P6a with the standard MTP problem settings but also has the following additional properties:

(P8) All instances are observed during training ($|\mathcal{X}| = n$).
(P9) No side information is available for instances, thus we identify them with natural numbers ($\mathcal{X} = \{1, \dots, n\}$).

In the matrix completion setting side information for instances and targets is missing, so the only achievable task is to complete the missing scores between instances and targets that are already observed during training. In these cases, matrix factorization methods utilize the structure of the score matrix $\mathbf{Y}$ in order to make predictions.

Definition 10 The **hybrid matrix completion** setting is an instance of the MTP framework that updates all the properties of the standard matrix completion setting:

(P4*) Novel targets are expected at prediction time ($|\mathcal{T}| = m^* > m$).
(P5*) Side information is available for the targets ($\mathcal{T} = \{\mathbf{t}_1, \dots, \mathbf{t}_m\}$) in the form of structured, hierarchical or feature representations.
(P8*) Novel instances are expected at prediction time ($|\mathcal{X}| = n^* > n$).
(P9*) Side information is available for the targets ($\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$) in the form of structured, hierarchical or feature representations.

Hybrid matrix completion extends the standard matrix completion setting by generalizing to novel instances and targets using the structure of the score matrix as well as side information. Both versions of the matrix completion method have been considered with great success in areas such as recommender systems, social network analysis and biological network inference. This wide adoption has also resulted in the establishment of new terms like collaborative filtering and link prediction.

Appendix B: Experimental setup and hyper-parameters

In the following section, we detail the methods we are comparing with, the datasets we use in a per MTP problem setting manner, and the hyperparameter space we explore. Similar to other papers in the area of deep learning, we decided to complete all the experiments using a train-test (75–25%) split. In papers predating the deep learning era, 5-fold or 10-fold cross-validation is usually the standard, but this has become less popular recently for problems that involve feature learning. Especially in many of the older multi-label classification papers, which often analyze low-dimensional datasets, cross-validation is often used. The choice for train-test splitting led us to rerun all the experiments for the methods we are comparing with. As a consequence we could only compare to methods that have an implementation available online.

For some of the methods included in our comparison, we performed hyperparameter optimization through a grid search, and for others, we used the default values included in their implementation. In the cases where such optimization was performed, we created an internal validation set to facilitate the process. The same strategy was used to perform early stopping in the training process of the DeepMTP framework, so that we eliminate any chance of information leaking from the original test set.

Multi-label classification

- Multilabel k Nearest Neighbours:

  - Number of neighbours of each input instance to take into account (k): [1–10]
  - Smoothing parameter (s): [0.5, 0.7, 1]

- Binary Relevance (support vector machine):

  - Regularization parameter (C): [0.01, 0.1, 1, 10, 100]
  - Kernel: [linear, rbf]

- Binary Relevance (Multilayer Perceptron):

  - Learning rate: [0.001, 0.01, 0.1]
  - Solver: [stochastic gradient decent, adam]
  - hidden sizes: [(32), (64), (128), (256), (512)]

- Multilayer Perceptron:

  - Learning rate: [0.001, 0.01, 0.1]
  - Solver: [stochastic gradient decent, adam]
  - hidden sizes: [(32), (64), (128), (256), (512), (32,32), (64,64), (128,128), (256,256), (512,512)]

- Multioutput Decision Tree Classifier:

  - Split criterion: [gini, random]
  - Splitting strategy: [best, random]
  - The minimum number of samples required to be at a leaf node: [1, 2, 3, 4, 5]

- Ensemble of Classifier Chains (ECC(support vector machine)):

  - size of ensemble: 50
  - Regularization parameter (C): [0.01, 0.1, 1, 10, 100]

- Kernel: [linear, rbf]

Multivariate regression

- Support Vector Regressor/target:
  - Regularization parameter (C): [0.01, 0.1, 1, 10, 100]
- Kernel Ridge Regressor/target:
  - Regularization strength (a): [0.01, 0.1, 1, 10, 100]
  - Gamma parameter: [0.01, 0.1, 1, 10, 100]
  - kernel: rbf
- Multilayer Perceptron:
  - Learning rate: [0.001, 0.01, 0.1]
  - Solver: [stochastic gradient decent, adam]
  - hidden sizes: [(32), (64), (128), (256), (512), (32,32), (64,64), (128,128), (256,256), (512,512)]
- Multioutput Decision Tree Regressor:
  - Split criterion: [mse, mae]
  - Splitting strategy: [best, random]
  - The minimum number of samples required to be at a leaf node: [1, 2, 3, 4, 5]
- Ensemble of Regressor Chains (ERC (support vector regressor)):
  - Regularization parameter (C): [0.01, 0.1, 1, 10, 100]

Hierarchical multi-label classification

- ML-GCN: used the provided default parameters

Matrix completion

Used the default parameters provided in Microsoft (2018) for the following methods:

- Matrix factorization approach by alternating least squares (MF-ALS)
- Fastai
- Riemannian Low-rank Matrix Completion (rlrmc)
- Wide & deep

Multi-task learning

- majority voting: no parameters to tune
- single-target Resnet: used the default parameters

Dyadic prediction

- eBICT:

  - number of trees: 200

Availability of data and materials The data used for the experiments are available online, see Sect. 6 for more details.

Code availability The code used to run the experiments can be found on github (https://github.ugent.be/diliadis/deepMTP_comparisons). The implementation of deepMTP will also be uploaded to the same repository in the near future.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Baltruschat, I. M., Nickisch, H., Grass, M., Knopp, T., & Saalbach, A. (2019). Comparison of deep learning approaches for multi-label chest X-ray classification. *Scientific Reports, 9*(1), 1–10.

Baumel, T., Nassour-Kassis, J., Cohen, R., Elhadad, M., & Elhadad, N. (2018). Multi-label classification of patient notes a case study on ICD code assignment. In *AAAI workshop on health intelligence*.

Bielza, C., Li, G., & Larranaga, P. (2011). Multi-dimensional classification with Bayesian networks. *International Journal of Approximate Reasoning, 52*(6), 705–727.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1993). Signature verification using a "Siamese" time delay neural network. *Advances in Neural Information Processing Systems, 6*, 737–744.

Cerri, R., Barros, R. C., & De Carvalho, A. C. (2014). Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences, 80*(1), 39–56.

Chen, G., Ye, D., Xing, Z., Chen, J., & Cambria, E. (2017). Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *2017 International joint conference on neural networks (IJCNN)* (pp. 2377–2383). IEEE.

Chen, H., Miao, S., Xu, D., Hager, G. D., & Harrison, A. P. (2019). Deep hierarchical multi-label classification of chest X-ray images. In *International conference on medical imaging with deep learning* (pp. 109–120). PMLR.

Chen, K., & Salman, A. (2011). Extracting speaker-specific information with a regularized Siamese deep network. In *NIPS* (vol. 201, pp. 298–306).

Chen, S. F., Chen, Y. C., Yeh, C. K., & Wang, Y. C. (2018). Order-free RNN with visual attention for multi-label classification. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32).

Chen, Z. M., Wei, X. S., Wang, P., & Guo, Y. (2019). Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5177–5186).

Chicco, D. (2021). Siamese neural networks: An overview. In *Artificial neural networks* (pp. 73–94).

Dacrema, M. F., Boglio, S., Cremonesi, P., & Jannach, D. (2021). A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems (TOIS), 39*(2), 1–49.

Das, A., Yenala, H., Chinnakotla, M., & Shrivastava, M. (2016). Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers) (pp. 378–387).

De'Ath, G. (2002). Multivariate regression trees: A new technique for modeling species-environment relationships. *Ecology, 83*(4), 1105–1117.

Deldjoo, Y., Dacrema, M. F., Constantin, M. G., Eghbal-Zadeh, H., Cereda, S., Schedl, M., et al. (2019). Movie genome: Alleviating new item cold start in movie recommendation. *User Modeling and User-Adapted Interaction, 29*(2), 291–343.

Dembczyński, K., Waegeman, W., Cheng, W., & Hüllermeier, E. (2010). Regret analysis for performance metrics in multi-label classification: The case of Hamming and subset zero-one loss. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 280–295). Springer.

Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., & Zhang, F. (2017). A hybrid collaborative filtering model with deep structure for recommender systems. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 31).

Du, J., & Xu, Y. (2017). Hierarchical deep neural network for multivariate regression. *Pattern Recognition, 63*, 149–157.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision, 88*(2), 303–338.

Frey, B. J., & Jojic, N. (2005). A comparison of algorithms for inference and learning in probabilistic graphical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(9), 1392–1416.

Fu, M., Qu, H., Yi, Z., Lu, L., & Liu, Y. (2018). A novel deep learning-based collaborative filtering model for recommendation system. *IEEE Transactions on Cybernetics, 49*(3), 1084–1096.

Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., et al. (2012). Chembl: A large-scale bioactivity database for drug discovery. *Nucleic Acids Research, 40*(D1), D1100–D1107.

Gong, Y., Jia, Y., Leung, T., Toshev, A., & Ioffe, S. (2013). Deep convolutional ranking for multilabel image annotation. arXiv preprint arXiv:1312.4894.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1). Cambridge: MIT Press.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The week data mining software: An update. *ACM SIGKDD Explorations Newsletter, 11*(1), 10–18.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

He, X., Du, X., Wang, X., Tian, F., Tang, J., & Chua, T. S. (2018). Outer product-based neural collaborative filtering. arXiv preprint arXiv:1808.03912.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (pp. 173–182).

Hoffer, E., & Ailon, N. (2015). Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition* (pp. 84–92). Springer.

Hua, Y., Mou, L., & Zhu, X. X. (2020). Relation network for multilabel aerial image classification. *IEEE Transactions on Geoscience and Remote Sensing, 58*(7), 4558–4572.

Huang, W., Chen, E., Liu, Q., Chen, Y., Huang, Z., Liu, Y., Zhao, Z., Zhang, D., & Wang, S. (2019). Hierarchical multi-label text classification: An attention-based recurrent network approach. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 1051–1060).

Huynh, D., & Elhamifar, E. (2020). A shared multi-attention framework for multi-label zero-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8776–8786).

Jain, P., Netrapalli, P., & Sanghavi, S. (2013). Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on theory of computing* (pp. 665–674).

Jia, B. B., & Zhang, M. L. (2020). Multi-dimensional classification via stacked dependency exploitation. *Science China Information Sciences, 63*(12), 1–14.

Jin, B., Yang, H., Xiao, C., Zhang, P., Wei, X., & Wang, F. (2017). Multitask dyadic prediction and its application in prediction of adverse drug-drug interaction. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 31).

Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., et al. (2021). Pubchem in 2021: New data content and improved web interfaces. *Nucleic Acids Research, 49*(D1), D1388–D1395.

Kimura, K., Sun, L., & Kudo, M. (2017). Mlc toolbox: A matlab/octave library for multi-label classification. arXiv preprint arXiv:1704.02592.

Kumar, P., Grewal, M., & Srivastava, M. M. (2018). Boosted cascaded convents for multilabel classification of thoracic diseases in chest radiographs. In *International conference image analysis and recognition* (pp. 546–552). Springer.

Lafferty, J., Mccallum, A., & Pereira, F. (2001). *Conditional random fields: Probabilistic models for segmenting and labeling sequence data* (pp. 282–289).

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755). Springer.

Liu, J., Chang, W. C., Wu, Y., & Yang, Y. (2017). Deep learning for extreme multi-label text classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval* (pp. 115–124).

Liu, S., Chen, C., Lu, Y., Ouyang, F., & Wang, B. (2018). An interactive method to improve crowdsourced annotations. *IEEE Transactions on Visualization and Computer Graphics, 25*(1), 235–245.

Liu, S., Johns, E., & Davison, A. J. (2019). End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1871–1880).

Liu, X., Liu, W., Mei, T., & Ma, H. (2017). Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance. *IEEE Transactions on Multimedia, 20*(3), 645–658.

Liu, Y., Qiu, S., Zhang, P., Gong, P., Wang, F., Xue, G., & Ye, J. (2017). Computational drug discovery with dyadic positive-unlabeled learning. In *Proceedings of the 2017 SIAM international conference on data mining* (pp. 45–53). SIAM.

Madjarov, G., Kocev, D., Gjorgjevikj, D., & Džeroski, S. (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern Recognition, 45*(9), 3084–3104.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., Zamparelli, R., et al. (2014). A sick cure for the evaluation of compositional distributional semantic models. In *Lrec* (pp. 216–223). Reykjavik.

McCullagh, P., & Nelder, J. A. (2019). *Generalized linear models*. London: Routledge.

Melki, G., Cano, A., Kecman, V., & Ventura, S. (2017). Multi-target support vector regression via correlation regressor chains. *Information Sciences, 415*, 53–69.

Menon, A. K., & Elkan, C. (2010). A log-linear model with latent features for dyadic prediction. In *2010 IEEE international conference on data mining* (pp. 364–373). IEEE.

Menon, A. K., & Elkan, C. (2011). Link prediction via matrix factorization. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 437–452). Springer.

Microsoft (2018). recommenders. https://github.com/microsoft/recommenders.

Mishra, A., Krishna Reddy, S., Mittal, A., & Murthy, H. A. (2018). A generative model for zero shot learning using conditional variational autoencoders. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 2188–2196).

Misra, I., Shrivastava, A., Gupta, A., & Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3994–4003).

Mueller, J., & Thyagarajan, A. (2016). Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 30).

Nam, J., Mencía, E. L., Kim, H. J., & Fürnkranz, J. (2017). Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 5419–5429).

Norouzi, M., Mikolov, T., Bengio, S., Singer, Y., Shlens, J., Frome, A., Corrado, G. S., & Dean, J. (2013). Zero-shot learning by convex combination of semantic embeddings. arXiv preprint arXiv:1312.5650.

Pan, J., Liu, S., Sun, D., Zhang, J., Liu, Y., Ren, J., Li, Z., Tang, J., Lu, H., Tai, Y. W., et al. (2018). Learning dual convolutional neural networks for low-level vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3070–3079).

Papagiannopoulou, C., Miralles, D. G., Demuzere, M., Verhoest, N. E., & Waegeman, W. (2018). Global hydro-climatic biomes identified via multitask learning. *Geoscientific Model Development, 11*(10), 4139–4153.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research, 12*, 2825–2830.

Pitt, M. A., Johnson, K., Hume, E., Kiesling, S., & Raymond, W. (2005). The buckeye corpus of conversational speech: Labeling conventions and a test of transcriber reliability. *Speech Communication, 45*(1), 89–95.

Pliakos, K., & Vens, C. (2019). Network inference with ensembles of bi-clustering trees. *BMC Bioinformatics, 20*(1), 1–12.

Pliakos, K., Vens, C., & Tsoumakas, G. (2019). Predicting drug-target interactions with multi-label classification and label partitioning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

Qi, D., & Roe, B. E. (2016). Household food waste: Multivariate regression and principal components anal-ysis of awareness and attitudes among us consumers. *PloS One, 11*(7), e0159250.

Raj, A., Shah, N. A., Tiwari, A. K., & Martini, M. G. (2020). Multivariate regression-based convolutional neural network model for fundus image quality assessment. *IEEE Access, 8*, 57810–57821.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2009). Classifier chains for multi-label classification. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 254–269). Springer.

Read, J., Reutemann, P., Pfahringer, B., & Holmes, G. (2016). Meka: A multi-label/multi-target extension to weka. *JMLR, 17*, 1–5.

Rendle, S., Krichene, W., Zhang, L., & Anderson, J. (2020). Neural collaborative filtering vs. matrix factori-zation revisited. In *Fourteenth ACM conference on recommender systems* (pp. 240–248).

Rifaioglu, A. S., Nalbat, E., Atalay, V., Martin, M. J., Cetin-Atalay, R., & Doğan, T. (2020). Deepscreen: High performance drug-target interaction prediction with convolutional neural networks using 2-d structural compound representations. *Chemical Science, 11*(9), 2531–2557.

Rokach, L., Schclar, A., & Itach, E. (2014). Ensemble methods for multi-label classification. *Expert Systems with Applications, 41*(16), 7507–7523.

Romera-Paredes, B., & Torr, P. (2015). An embarrassingly simple approach to zero-shot learning. In *Inter-national conference on machine learning* (pp. 2152–2161). PMLR.

Ryoo, M., Kim, K., & Yang, H. (2018). Extreme low resolution activity recognition with multi-Siamese embedding learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32).

Schäfer, D., & Hüllermeier, E. (2015). Dyad ranking using a bilinear Plackett–Luce model. In *Joint Euro-pean conference on machine learning and knowledge discovery in databases* (pp. 227–242). Springer.

Schreiber, J., Durham, T., Bilmes, J., & Noble, W. S. (2020). Avocado: A multi-scale deep tensor factoriza-tion method learns a latent representation of the human epigenome. *Genome Biology, 21*(1), 1–18.

Sener, O., & Koltun, V. (2018). Multi-task learning as multi-objective optimization. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Advances in neural informa-tion processing systems (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/432aca3a1e345e339f35a30c8f65edce-Paper.pdf

Shan, H., & Banerjee, A. (2010). Generalized probabilistic matrix factorizations for collaborative filtering. In *2010 IEEE international conference on data mining* (pp. 1025–1030). IEEE.

Shao, J., & Qian, Y. (2019). Three convolutional neural network models for facial expression recognition in the wild. *Neurocomputing, 355*, 82–92.

Shatkay, H., Pan, F., Rzhetsky, A., & Wilbur, W. J. (2008). Multi-dimensional classification of biomedi-cal text: Toward automated, practical provision of high-utility text to diverse users. *Bioinformatics, 24*(18), 2086–2093.

Strub, F., Gaudel, R., & Mary, J. (2016). Hybrid recommender system based on autoencoders. In *Proceed-ings of the 1st workshop on deep learning for recommender systems* (pp. 11–16).

Szymański, P., & Kajdanowicz, T. (2017). A scikit-based Python environment for performing multi-label classification. ArXiv e-prints.

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM), 3*(3), 1–13.

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2010). Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering, 23*(7), 1079–1089.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., & Vlahavas, I. (2011). Mulan: A JAVA library for multi-label learning. *The Journal of Machine Learning Research, 12*, 2411–2414.

Van Peer, G., De Paepe, A., Stock, M., Anckaert, J., Volders, P. J., Vandesompele, J., De Baets, B., & Wae-geman, W. (2017). miSTAR: miRNA target prediction through modeling quantitative and qualitative mirna binding site information in a stacked model structure. *Nucleic Acids Research, 45*(7), e51–e51.

Waegeman, W., Dembczyński, K., & Hüllermeier, E. (2019). Multi-target prediction: A unifying view on problems and methods. *Data Mining and Knowledge Discovery, 33*(2), 293–324.

Wang, D., Shi, L., & Heng, P. A. (2009). Automatic detection of breast cancers in mammograms using structured support vector machines. *Neurocomputing, 72*(13–15), 3296–3302.

Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., & Xu, W. (2016). CNN-RNN: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pat-tern recognition* (pp. 2285–2294).

Wang, L., Li, Y., Huang, J., & Lazebnik, S. (2018). Learning two-branch neural networks for image-text matching tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 41*(2), 394–407.

Wang, X., He, X., Wang, M., Feng, F., & Chua, T. S. (2019). Neural graph collaborative filtering. In *Pro-ceedings of the 42nd international ACM SIGIR conference on research and development in informa-tion retrieval* (pp. 165–174).

Wehrmann, J., Cerri, R., & Barros, R. (2018). Hierarchical multi-label classification networks. In *International conference on machine learning* (pp. 5075–5084). PMLR.

Wei, J., He, J., Chen, K., Zhou, Y., & Tang, Z. (2017). Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications, 69*, 29–39.

Wei, S., Zheng, X., Chen, D., & Chen, C. (2016). A hybrid approach for movie recommendation via tags and ratings. *Electronic Commerce Research and Applications, 18*, 83–94.

Wei, Y., Xia, W., Lin, M., Huang, J., Ni, B., Dong, J., et al. (2015). Hcp: A flexible CNN framework for multi-label image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 38*(9), 1901–1907.

Wishart, D. S., Knox, C., Guo, A. C., Shrivastava, S., Hassanali, M., Stothard, P., Chang, Z., & Woolsey, J. (2006). DrugBank: A comprehensive resource for in silico drug discovery and exploration. *Nucleic Acids Research*, *34*(suppl_1), D668–D672.

Wu, X., Shi, B., Dong, Y., Huang, C., & Chawla, N. (2018). Neural tensor factorization. arXiv preprint arXiv:1802.04416.

Wu, X., Shi, B., Dong, Y., Huang, C., & Chawla, N. V. (2019). Neural tensor factorization for temporal interaction learning. In *Proceedings of the Twelfth ACM international conference on web search and data mining* (pp. 537–545).

Xu, S., An, X., Qiao, X., Zhu, L., & Li, L. (2013). Multi-output least-squares support vector regression machines. *Pattern Recognition Letters, 34*(9), 1078–1084.

Yan, Z., Liu, W., Wen, S., & Yang, Y. (2019). Multi-label image classification by feature attention network. *IEEE Access, 7*, 98005–98013.

Yeh, C. K., Wu, W. C., Ko, W. J., & Wang, Y. C. F. (2017). Learning deep latent space for multi-label classification. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 31).

Yi, D., Lei, Z., Liao, S., & Li, S. Z. (2014). Deep metric learning for person re-identification. In 2014 22nd international conference on pattern recognition (pp. 34–39). IEEE.

Yih, W. t., Toutanova, K., Platt, J. C., & Meek, C. (2011). Learning discriminative projections for text similarity measures. In *Proceedings of the fifteenth conference on computational natural language learning* (pp. 247–256).

Yu, Y., Pedrycz, W., & Miao, D. (2014). Multi-label classification by exploiting label correlations. *Expert Systems with Applications, 41*(6), 2989–3004.

Zhang, L., Shah, S. K., & Kakadiaris, I. A. (2017). Hierarchical multi-label classification using fully associative ensemble learning. *Pattern Recognition, 70*, 89–103.

Zhang, M. L., & Zhou, Z. H. (2007). Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition, 40*(7), 2038–2048.

Zhang, S. X., & Gales, M. J. (2011). Structured support vector machines for noise robust continuous speech recognition. In *Twelfth annual conference of the international speech communication association*.

Zhang, W., Yan, J., Wang, X., & Zha, H. (2018). Deep extreme multi-label learning. In *Proceedings of the 2018 ACM on international conference on multimedia retrieval* (pp. 100–107).

Zhao, L., Lu, Z., Pan, S. J., Yang, Q., & Xu, W. (2016). Matrix factorization+ for movie recommendation. In *IJCAI* (pp. 3945–3951).

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., & Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 1529–1537).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.