Multi-task pre-training of deep neural networks for digital pathology

Romain Mormont, Pierre Geurts, and Raphaël Marée

***Abstract*—In this work, we investigate multi-task learning as a way of pre-training models for classification tasks in digital pathology. It is motivated by the fact that many small and medium-size datasets have been released by the community over the years whereas there is no large scale dataset similar to ImageNet in the domain. We first assemble and transform many digital pathology datasets into a pool of 22 classification tasks and almost 900k images. Then, we propose a simple architecture and training scheme for creating a transferable model and a robust evaluation and selection protocol in order to evaluate our method. Depending on the target task, we show that our models used as feature extractors either improve significantly over ImageNet pre-trained models or provide comparable performance. Fine-tuning improves performance over feature extraction and is able to recover the lack of specificity of ImageNet features, as both pre-training sources yield comparable performance.**

***Index Terms*—deep learning, multi-task learning, digital pathology, transfer learning**

I. Introduction

Recent advances in deep learning have moved forward the field of computer vision. Those developments have been made possible by the availability of large datasets, efficient computing resources and algorithms. These successes have inspired communities to apply deep learning techniques in other fields of application (astronomy, medicine, geography, etc.) where images are prevalent but classical computer vision methods yield unsatisfying performance.

Digital pathology (DP), a domain of medical imaging that focuses on the analysis of large digitized glass slide images (*a.k.a.* whole-slide-images, WSI) containing tissue and cell samples, is no exception. Several groups have applied deep learning on such images for various research and clinical tasks, including cell detection, counting and classification, as well as tissue segmentation. Although deep learning has shown promising results, several difficulties hamper the usefulness of these approaches in practice. One of the main issues is the scarcity of the data [12]–[15]. Deep learning algorithms are indeed data-hungry and the number and scale of available datasets is usually much lower in digital pathology than in the natural image domain where these methods have shown the most benefit [16]. Some of the reasons are the cost and time of the annotation process, which requires the participation of medical experts, but also privacy concerns, which prevent researchers and hospitals from sharing patient data. Moreover, digital pathology images are subject to several sources of variability specific to the process of acquiring samples and turning them into images (tissue preparation protocols including various staining procedures, scanning artefacts, etc.).

When tackling a new digital pathology classification problem, re-using deep learning techniques developed in other fields does not usually work off-the-shelf because of data scarcity. Therefore, one common approach that has been considered to overcome this issue is transfer learning. The core idea is to pre-train a model on a large dataset (the source task), and then somehow transfer the learned knowledge to facilitate training on a second dataset (the target task). As the source task must be a large dataset, the most common choice is using ImageNet, a classification dataset containing more than 1 million natural images organized into 1000 classes, as a source. Although ImageNet images are very dissimilar to digital pathology images, it has been shown that transferring from the former can still boost performance of deep learning methods on the latter [9], [17].

It has also been shown, however, that transfer learning works best when the target task is similar to the source task [18]. Whereas task similarity is hard to define formally, it is clear that the ImageNet task is not similar to any digital pathology task. Therefore, this question arises: could we get even better performance from transfer learning by using digital pathology pre-trained models instead of an ImageNet one ? Some works, e.g., [17], [19]–[21], have advanced that domain-specific pre-training is indeed beneficial but, to the best of our knowledge, there is no in-depth study that attempted to answer this question in digital pathology. The main obstacle preventing this question to be answered is the lack of a large and versatile dataset like ImageNet. However, the digital pathology community has made available many small and medium size datasets through challenges and publications during the last years. This motivates to consider the use of multi-task learning [22] (MTL), a subfield of supervised learning which focuses on methods that solve several tasks simultaneously. Learning several tasks at once presents several advantages such as implicitly regularizing training, therefore helping convergence and reducing overfitting. Provided that enough relevant tasks are available, it makes MTL a great candidate to cope with the data scarcity problem of digital pathology.

Therefore, in this work, we investigate MTL as a way

Submitted for review on 30 November 2019. Accepted for publication on 1 May 2020. Romain Mormont, Pierre Geurts and Raphaël Marée are all affiliated with the Department of Electrical Engineering and Computer Science of the University of Liège, 4000, Liège, Belgium. (e-mail: r.mormont@uliege.be, p.geurts@uliege.be and raphael.maree@uliege.be).

TABLE I
DATASETS THAT WERE USED FOR MULTI-TASK PRE-TRAINING. CLF, DET AND SEG RESPECTIVELY STAND FOR *classification*, *detection* AND *segmentation*. H&E, IHC AND M3C RESPECTIVELY STAND FOR *hematoxylin and eosin*, *immunohistochemistry* AND *Masson's trichrome*. *Images* AND *Classes* COLUMNS GIVE THE NUMBER OF IMAGES AND CLASSES OF THE FINAL (POSSIBLY TRANSFORMED) TASK FOR THIS DATASET.

Name	Type	Task	Organ & pathology	Stains	Images	Classes
MITOS-ATYPIA 14 [1]	DET	Detection of mitosis and grading nuclear atypia	breast cancer	H&E	64873	3
Warwick CRC [2]	DET	Detection and classification of nuclei	colorectal cancer	H&E	2500	2
Janowczyk1 [3]	SEG	Cell nuclei segmentation	breast cancer	H&E	31725	2
Janowczyk2 [3]	SEG	Identification of epithelium and stroma	breast cancer	H&E	3402	2
Janowczyk5 [3]	DET	Detection of mitosis	breast cancer	H&E	24870	2
Janowczyk6 [3]	CLF	Patch classification for WSI segmentation	breast, invasive ductal carci.	H&E	277524	2
Janowczyk7 [3]	CLF	Identification of lymphoma subtypes	breast, lymphoma	H&E	2244	3
Stroma LBP [4]	CLF	Identification of epithelium and stroma	colorectal cancer	IHC	2313	2
TUPAC2016 Mitosis [5]	DET	Detection of mitosis	breast cancer	H&E	77853	2
BACH18 Micro [6]	CLF	Predominant cancer type classification	breast cancer	H&E	4800	4
Camelyon16 [7]	SEG	Detection of lymph nodes metastases	breast cancer	H&E	292226	2
UMCM Colorectal [8]	CLF	Tissue type classification	colorectal cancer	H&E	5000	8
Necrosis [9]	CLF	Necrosed vs healthy tissue	breast cancer	IHC	882	2
ProliferativePattern [9]	CLF	Prolif. vs non-prolif. classification	thyroid cancer	Diff-Quick	1857	2
CellInclusion [9]	CLF	Cell inclusion vs healthy cell classification	thyroid cancer	Diff-Quick	3637	2
MouseLba [9]	DET	Cell classification in bronchoalveolar lavage	lung cancer	MGG	4284	8
HumanLba [9]	DET	Cell classification in bronchoalveolar lavage	lung cancer	MGG	5420	9
Lung [9]	CLF	Tissue subtype classification	lung	H&E	6331	10
Glomeruli [10]	CLF	Glomeruli recognition	kidney	M3C	29213	2
Breast1 [9]	CLF	Segmentation of cancer tissue	breast cancer	H&E	23032	2
Breast2 [9]	CLF	Segmentation of cancer tissue	breast cancer	H&E	17523	2
Bone marrow [11]	CLF	Cell type classification	bone marrow	H&E	1291	9
				Total	882800	81

of pre-training neural networks for digital pathology. Our main contributions are as follows. **(1)** We have collected, assembled and transformed heterogeneous digital pathology datasets into a large versatile pool of classification datasets featuring 22 tasks, 81 classes and almost 900k images (see Section III). **(2)** We have developed a multi-task architecture and a corresponding training scheme for creating a transferable model from these 22 tasks (see Sections IV-A to IV-C). **(3)** We have developed a robust validation protocol based on a leave-one-task-out scheme for evaluating the transfer performance of our models compared to other approaches (see Sections IV-D to IV-F). **(4)** We have evaluated the performance of the resulting multi-task pre-trained models compared to ImageNet ones, both when pretrained models are used as direct feature extractors and when they are fine-tuned for each target task. We have also compared our approach to a model trained from scratch without any transfer, as well as to a MTL model trained including the target dataset (see Section V). **(5)** Our implementation and multi-task pre-trained models are available on GitHub[2].

II. RELATED WORK

Transfer learning is not a recent field of research [23] but has grown in popularity with deep learning as it has been shown that features learned on a source task by a neural network could be transferred to a possibly unrelated target task [18], [24], [25]. The success of this approach is mostly due to the possibility to use the large and versatile ImageNet [16] dataset as a source task [26]. Medical imaging and digital pathology communities have therefore studied and used transfer learning [9], [27]–[30] as it provides a way of coping with data scarcity. Those works have explored and evaluated different transfer techniques mostly using ImageNet as a source task. The current consensus is that transfer is helpful in most cases and should be considered when tackling a new task. More recently, several works have focused on transferring a model pre-trained on medical, biology or digital pathology datasets. This is motivated by the fact that one can expect better performance from transfer learning when target and source task are close or related [18]. In [19], Khan *et al.* pre-train an InceptionV3 network [31] on a custom dataset generated from Camelyon16 [7] and then transfer the resulting model to a prostate cancer classification task. They show that their pre-trained model outperforms both training from scratch and using an ImageNet pre-trained model. Medela *et al.* [20] also make use of transfer learning between two digital pathology tasks but use a different pre-training approach. Indeed, they train a siamese network to distinguish different parts of colorectal tissues. The network is then transferred as a feature extractor on the target task (tumour classification). Shang *et al.* [17] use several datasets (including some unrelated to their target task such as *Dogs vs. cats*) and compare ImageNet and domain-specific pre-training in order to tackle colonoscopy image classification. They also show that pre-training on domain-specific data yield superior performance compared to using ImageNet. Kraus *et al.* [21] train a custom deep neural architecture, DeepLoc, for classifying protein subcellular localization in budding yeast. Then, they assess the transferability of their pre-trained DeepLoc by fine-tuning it on different image sets including unseen classes and show that the pre-training is indeed beneficial.

Independently, multi-task learning [22] has been applied with great successes for a wide-range of application. The success of MTL is notably due to the fact that leveraging several tasks and/or datasets alleviates the need for large amounts of data. Moreover, training in multi-task has regularization effect preventing the model to overfit a particular task therefore

[2]https://github.com/waliens/multitask-dipath

yielding a better generalizing model. The modularity of neural networks also allows to embed multi-task specific components hence facilitating its application to deep learning [22], [32]. There are many ways how MTL can be implemented within deep learning with, for instance, architecture tricks [33], [34] or weight sharing [32].

Multi-task learning has been applied to medical imaging. Samala *et al.* [35] jointly train a classifier on three mammography image datasets (digitized screen-film and digital mammograms) and compare it to single-task training and transfer learning. They show that multi-task trained network generalizes better than the single-task one. Zhang *et al.* [36] use transfer and multi-task learning to derive image features from Drosophila gene expression. MTL has also been applied more specifically to digital pathology. Pan *et al.* [37] apply MTL for breast cancer classification by using a classification loss and a verification loss. The role of the latter is to ensure that features produced by the network differ for images of different classes. Arvaniti *et al.* [38] use both weak and strong supervision at once to classify prostate cancer. Shang *et al.* (mentioned earlier) also evaluate multi-task learning which is the best performing approach on their target task. However, they suggest that more experiments would have to be carried out to assess whether their conclusions are generalizable.

Our work lies at the crossroad of multi-task and transfer learning and differs from the previously presented works mostly on the objective. Indeed, we do not use MTL nor transfer learning for solving a specific task but rather to pre-train a versatile network to be transferred to new tasks.

III. Data

In order to build our pool of tasks, we have collected publicly available datasets (see Table I) from as many sources as possible. We have also leveraged the Cytomine [39] platform to collect additional datasets annotated by our collaborators. Some publicly available datasets are missing from our pool because either they could not be converted into a relevant classification problem (e.g. KimiaPath24 [40], Janowczyk tutorials 3 & 4 [3]) or we could not actually obtain them from the authors (dead link on download page or datasets not released yet [41]). Most datasets in our pool are H&E stained images of human breast cancer but some other organs, pathology and stains are represented, as well as cytology samples, and animal tissues. Also missing in the pool is the BreakHis [42] dataset which was kept aside during the development of multi-task training protocol for final model evaluation and comparison to other transfer approaches published using the dataset (see Supplementary Section E).

For the collected datasets to be used in a multi-task classification setting, some dataset-specific pre-processing procedures had to be executed on most of them. Applying those procedures, we have constructed a pool of 22 classification tasks which contains both binary and multiclass classification problems. The different pre-processing are detailed in Supplementary Section A where selected samples of the final tasks are also provided. Whereas we have tried to avoid intra-dataset class imbalance, there is major inter-dataset imbalance regarding the number of images: the smallest dataset contains 882 images whereas the largest one contains almost 300k. However, we believe it is not an issue and can be made of minor significance by adopting an ad-hoc multi-task training protocol (see Section IV-B).

IV. Methods

In the following section, we present the training and evaluation protocols and the experiments we have carried out. Those experiments have two main objectives. The first is to evaluate how performance of multi-task and ImageNet pre-trained networks compare when transferred to a target task. The second is to better understand how various training hyperparameters and choices impact the transfer of a multi-task pre-trained network. The multi-task architecture and training are described in IV-A and IV-B. We present the different transfer techniques we have used in Section IV-C. We have developed a model evaluation and selection protocol which is described in Sections IV-D and IV-E whereas the various parameters we have chosen and/or evaluated as well as the experiments we have carried out are presented in Section IV-F.

Regarding notations, we consider a multi-task setting with a pool $\mathcal{P}$ of T classification tasks. Each task t_i has n_{t_i} training samples and its classification problem features C_{t_i} classes. We use θ_k to designate interchangeably a (part of a) network and its parameters. The notation $|\theta_k|$ designates the number of trainable parameters of the network θ_k. $\mathcal{B}$ represents the set containing all samples from a batch and the batch size is denoted by $B = |\mathcal{B}|$. $\mathcal{B}_{t_i} \subseteq \mathcal{B}$ is a set that contains all the samples from task t_i in batch $\mathcal{B}$.

A. Multi-task architecture

The structure of our multi-task neural network is similar to those of [17] and [34] and is guided by the objective of pre-training a network for transfer. Therefore, we have adopted the architecture presented in Figure 1. The to-be transferred network is shared for all tasks and denoted by θ_s. We attach a head θ_i to θ_s for each task t_i in the pool $\mathcal{P}$. The head θ_i is simply a fully connected layer of dimensions $f_s \times C_{t_i}$ where f_s is the number of features produced by θ_s. Using such a simple layer has the benefit of making the learning capacity of the heads much lower compared to the shared network (in our experiments $|\theta_s| \gg |\theta_i|, \forall i$), hence forcing θ_s to learn relevant features for all tasks. In each head, a softmax is attached after the fully connected layer for producing per-task predictions. When forwarding samples in the multi-task network, samples of a given task t_i are only routed through the head θ_i, which outputs predictions for those samples.

B. Multi-task training

Classical training choices have to be adapted for the multi-task setting. Regarding batch sampling, we have decided to interleave tasks at the sample level, meaning that a single batch can contain samples from different tasks. Indeed, we believe that if batches containing samples from only a single task were alternated, the network would not see a particular task

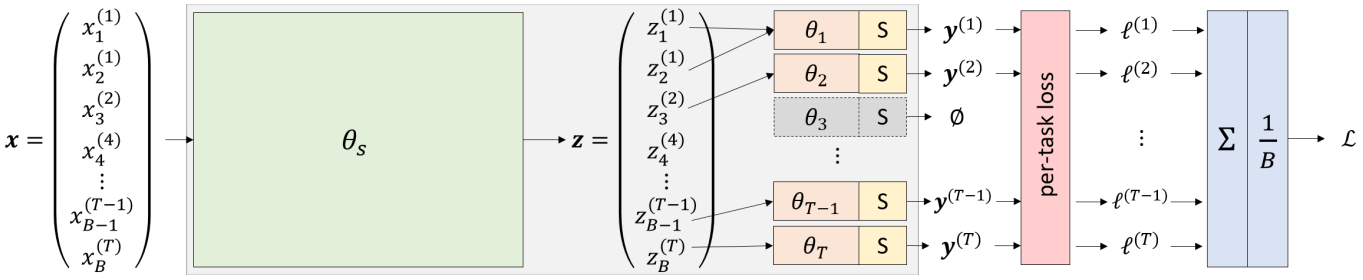


Fig. 1. Multi-task architecture. $\mathcal{L}$ is the multi-task loss (see Section IV-B) and S is a softmax layer. $x_j^{(i)}$ and $z_j^{(i)}$ designate respectively the j^{th} sample of the batch $\mathbf{x}$ and its corresponding features produced by θ_s. This sample belongs to task t_i. Features produced for samples of a given task t_i are routed to this task head θ_i. In this example, there is no sample for task 3 in the batch $\mathbf{x}$. Therefore, the corresponding head θ_3 is inactive (*i.e.* produces no output, no parameters update, no gradient computed) for this iteration.

for $T-1$ iterations, with T the number of tasks, which could make the training harder when T is large.

Given the imbalance in terms of number of images per task, a batch sampling procedure had to be carefully established. Indeed, a simple random sampling across all images would prevent the tasks with fewer images from being seen during training. To overcome this issue, we selected each image in a batch by first randomly sampling a task and then sampling an image from this task, thus giving equal weights to all tasks.

Regarding the training loss, we averaged the categorical cross-entropy over all batch samples taking into account their respective tasks. More precisely, the multi-task loss $\mathcal{L}$ is computed from a batch as:

$$\mathcal{L} = \frac{1}{B}\sum_{i=1}^{T} \ell^{(i)}, \;\; \ell^{(i)} = -\sum_{j=1}^{|\mathcal{B}_{t_i}|}\sum_{k=1}^{C_{t_i}} y_{j,k}^{(i)} \log \hat{y}_{j,k}^{(i)} \qquad (1)$$

where $\ell^{(i)}$ is the loss for the batch samples from task t_i.

When developing a multi-task algorithm, a crucial question is what should be shared between tasks. By reducing the capacity of the heads of our architecture, we wanted to enforce the training algorithm to find generic features in θ_t that work well for all tasks. It might be interesting however to provide a way to slightly relax this implicit constraint, with a hyperparameter. To this end, during training, whereas we train the network with a learning rate γ, we choose to train the heads with a potentially different learning rate given by $\gamma_h = \gamma \times \tau_\gamma$ where $\tau_\gamma \in \mathbb{R}_{\geq 0}$ is a multiplicative factor applied to the global learning rate. This new hyperparameter provides a way of tuning the specificity/genericity of the learned shared features. Indeed, $\tau_\gamma > 1$ makes the heads learning rate larger and gives therefore more flexibility for the heads to adapt, hence relieving the shared network from learning task-specific features. Taking $\tau_\gamma = 1$ results in using the same learning rate for the whole network.

As previously mentioned, each head θ_i is randomly initialized, whereas we initialize θ_s with ImageNet pre-trained weights as it has been shown that doing so accelerates convergence in a single-task setting [9]. However, it means that trained features of θ_s are followed directly by the random layers of the heads. This is known to hurt performance in a single-task transfer setting as reported in Yosinski *et al.* [18] and is aggravated in a multi-task setting. Indeed, during the first training iterations, the heads gradients will be relatively large and will work to turn each head weights from random to relevant with respect to its task. However, the resulting back-propagated gradients in the last layer of θ_s will be an average of the potentially contradictory signals coming from all the heads. In order to attenuate or eliminate this problem, a simple idea consists in making each head weights relevant to its task before training the whole network by running a warm up phase during which θ_s is frozen (*i.e.* no weights update, no batch normalization update) and only the heads θ_i are trained with a learning rate γ_w.

While preparing our experiments, we have noticed an issue with batch normalization [43] which is a consequence of the transfer learning settings. Indeed, it has been shown that using a batch normalization-equipped network with datasets across different domains can hurt performance [44], [45]. We have applied a simple procedure detailed in Supplementary Section B in order to correct the problem. Moreover, the fact that samples are routed in different heads during training requires careful treatment of the gradients after the forward phase. This question is discussed in Supplementary Section C.

C. Transferring a multi-task pre-trained network

In this work, we study the two classical approaches of network transfer, namely feature extraction and fine-tuning [13]. In both cases, θ_s is pre-trained on some source task(s), either ImageNet or several tasks simultaneously in the MTL setting. Feature extraction consists in using the pre-trained θ_s only to extract the feature vector it outputs for all images of the target task. The extracted features can then be used to learn a third-party classifier, a common choice being a linear SVM [9], [25]. Fine-tuning consists in further training θ_s on the target task. A fully connected layer and a softmax are attached to the shared network for generating the target task classes probabilities and the resulting network is trained using for instance stochastic gradient descent.

Both approaches have their own complementary advantages and drawbacks. As mentioned above, feature extraction uses a linear model which is very fast to train and makes it robust to overfitting when working with small target datasets. However, using fixed pre-trained features makes it possible that the features are not entirely suited for the target task (*e.g.* ImageNet vs. digital pathology), yielding suboptimal

performance. Fine-tuning does not suffer from this drawback as the whole network (or a part of it) is retrained on the target task. When using large capacity networks (*e.g.* ResNet or DenseNet), it allows the network to capture and learn task-specific features. This is however an issue when the target dataset is small because the large capacity of the network can lead to overfitting.

D. Evaluating transferability for hyperparameter tuning

Given a set of tasks available to pre-train a MTL model for future transfer, either by feature extraction or fine-tuning, a question left is how to tune the hyperparameters to train this model. Since we want to optimize the transferability of the model, rather than for example its average performance on the training tasks, we have to design a specific evaluation protocol and a specific metric to assess this transferability for each hyperparameter combination.

For this purpose, we have developed a leave-one-task-out (LOTO) cross-validation scheme, inspired from leave-one-out cross-validation. It consists in removing a set $\mathcal{T} \subset \mathcal{P}$ of one or more tasks from the training pool $\mathcal{P}$, training a multi-task model on $\mathcal{P} \setminus \mathcal{T}$ and then evaluating how the learnt models transfer to the tasks of $\mathcal{T}$. This operation can then be repeated for different $\mathcal{T}$ to increase the stability of the analysis. In our case, we have picked $\mathcal{T}$ to contain only one task when possible. However, it is important that tasks that are closely related are left out together during LOTO cross-validation to avoid leaking shared information between the related tasks during training. In our case, there are two pairs of datasets that are subject to this exception. The first is CellInclusion and ProliferativePattern which are different classification tasks coming from the same WSIs. The second is Breast1 and Breast2 which are the same classification tasks generated with different rules from the same expert annotations. Therefore, applying LOTO exhaustively in our settings leads to 20 possible left out sets $\mathcal{T}$.

The optimal hyperparameter combination might arguably depend on whether the MTL model will be used for feature extraction or fine-tuning. We have however solely used feature extraction performance as a proxy to evaluate transferability, mainly because we wanted to release a single MTL model for simplicity, but also to reduce the computational costs of our experiments. More precisely, given a left-out set $\mathcal{T}$ and one of its task $t \in \mathcal{T}$, we have evaluated transferability of a multi-task pre-trained network trained on $\mathcal{P} \setminus \mathcal{T}$ by using the resulting θ_s as a feature extractor on t. The training set of t was used to train the features classifier (*i.e.* a linear SVM, see Section IV-F for details) and the validation set was used to evaluate it. Transfer performance was evaluated by the accuracy (ACC) for multi-class classification tasks and the area under the receiver operating characteristic curve (ROC AUC) for binary classification tasks. To cope with the randomness induced by heads initialization and mini-batch sampling, each training of a MTL model was repeated with 5 different random seeds. Note that, at this stage, the test set of t was kept aside for future comparison of a selected multi-task pre-trained network and comparison to ImageNet transfer (see Section IV-E).

All the scores resulting from the same hyperparameters combination but different random seeds can be averaged and the resulting average performance can be used to assess transfer performance on a given task t. We need however to aggregate these scores over all (left-out) tasks to assess the overall transferability of a given hyperparameter combination. Averaging ACC and ROC AUC scores over tasks is in general not a good idea as these values depends on the task difficulty and are not directly comparable across tasks. We propose instead to aggregate rankings. More precisely, for each task, the combination of hyperparameters leading to the best model on average was assigned rank 1 and the worst was assigned the maximum rank. Applying this procedure to all our sets $\mathcal{T}$ produces a rank matrix where r_{ij} is the rank of the i^{th} combination of hyperparameters evaluated on the left-out task j. The best hyperparameter combination is then defined as the one that minimizes the average rank over all left-out tasks:

$$\overline{r}_i = \frac{1}{T_{\text{out}}} \sum_{j=1}^{T_{\text{out}}} r_{ij} \tag{2}$$

where T_{out} is the number of left-out tasks.

E. Final performance evaluation

Our main objective is to compare transfer from multi-task and ImageNet pre-trained networks. In principle, two nested LOTO cross-validation loops should be adopted to carry out such comparison: for each left-out task in the external LOTO CV loop, an additional internal LOTO CV loop should be run to find the optimal hyperparameter combination for training the MTL model to be transferred to the (external) left-out task. Using two nested loops would be however too expensive computationally[3]. We have instead adopted the following simplified scheme. A single LOTO cross-validation is run as described in the previous section. Given a left-out task t_k, we select the hyperparameter combination that minimizes the average rank but now excluding task t_k from the average computation (*i.e.* excluding r_{ik} from the calculation of $\overline{r}_i$ in Equation 2). All the models trained using this hyperparameters combination (*i.e.* one per seed) are transferred to the target task t_k using both transfer protocols (*i.e.* feature extraction or fine-tuning). The test set of t_k is used solely to evaluate the resulting transfer performance whereas the training and validation sets can be used by the transfer protocol, feature extraction or fine-tuning, for training and hyperparameter tuning (see Section IV-F).

Unlike with a true double LOTO CV loop, the training and validation sets of the left-out task t_k are used, in our simplified scheme, to train some of the MTL models that are transferred to the other tasks for computing the rankings of the hyperparameters combinations for these tasks. However, this is not expected to introduce any bias since the data from task t_k is neither used to decide on the optimal hyperparameter setting of the MTL model transferred to t_k itself (since r_{ik} is

[3]The simplified scheme we present hereafter has already yielded approximately 20k GPU hours of computation. This computation time would have been multiplied by 10 using two nested LOTO CV loops, which was impossible for us to carry out given our available computing resources.

TABLE II
THE MULTI-TASK TRAINING PARAMETERS EVALUATED USING THE CROSS-VALIDATION PROCEDURE. USING THE LOTO SCHEME, 240 TRAINED MODELS SHOULD BE TRANSFERRED TO EACH LEFT OUT DATASET. $\mathcal{H}$ IS THE SET OF ALL HYPERPARAMETERS COMBINATIONS.

Parameters		Values	Count
Learning rate (LR)	γ	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$	4
LR multiplier	τ_γ	$\{1, 5, 10\}$	3
Shared network	θ_s	$\{\text{ResNet50}, \text{DenseNet121}\}$	2
Warm up	w	$\{true, false\}$	2
Number of combinations $\lvert\mathcal{H}\rvert$			48
with random seeds			240

excluded from the computation of $\overline{r}_i$), nor to train this MTL model.

F. Hyperparameters settings and experiments

The hyperparameters we have studied and their evaluated values are listed in Table II. Parameters values and training choices were established based on early experiments which evaluated multi-task training stability and convergence. Regarding the selected range of learning rates, values higher than 10^{-3} resulted in very unstable or diverging trainings whereas values lower than 10^{-6} prevented convergence. As shared network θ_s, we have used two popular architectures ResNet50 [46] and DenseNet121 [47]. We have removed the fully connected layer of those networks and replaced it by a global average pooling. Moreover, we have loaded the networks with ImageNet pre-trained weights from PyTorch [48].

The multi-task network was trained for 50k iterations using batches of size 64 and SGD as optimizer using momentum set to its default value (*i.e.* 0.9), learning rate γ and heads learning rate multiplier τ_γ. Either the whole network was trained directly, or the heads were first warmed up for 5k iterations with learning rate $\gamma_w = 10^{-3}$ before the whole network was trained for 45k iterations.

Classical data augmentation and normalization have been applied to the input images. We have used ImageNet statistics for normalizing the images as early experiments have shown no significant improvement by normalizing with per-task statistics. As data augmentation, we have applied simple random vertical and horizontal flips as well as extraction of a random square crop (if the image is not square already).

When feature extraction was applied either during the evaluation or selection, we have used linear SVM [49] as feature classifier. Whenever a SVM classifier was trained, we have tuned the C regularization parameter among the following values $\left\{10^{-10}, 10^{-9}, ..., 10^{-1}, 1\right\}$ by 5-fold cross-validation. The tasks were splitted into folds based on the most relevant information available for the task (patient, slide or image).

Regarding fine-tuning, we have trained the network for 100 epochs on the training set of the target task and have tuned the learning rate among $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ and have selected the best epoch on the validation set. When transferring from our multi-task pre-trained models, we have performed the selection of the best multi-task models as explained in Section IV-D. Then, for each model trained with the best hyperparameters (*i.e.* one per seed), we have trained one model per fine-tuning hyperparameters combination. For ImageNet, the approach was slightly different as we had only one model to start from. In this case, we have used 5 different seeds for fine-tuning.

At this point, it is important to note that LOTO cross validation is quite demanding in terms of computing resources as, for all $\mathcal{T}$, all the combinations of hyperparameters and random seeds have to be evaluated. As indicated in Table II, 240 models would have to be trained per set $\mathcal{T}$ of excluded tasks which, given 22 tasks, yields 4800 multi-task trainings. Due to limited availability of computing resources, we had to reduce this number and have done so by reducing the number of left out tasks used in our analysis. In particular, we have kept 10 tasks in 8 sets: $\{\text{CellInclusion}, \text{ProliferativePattern}\}$, $\{\text{Breast1}, \text{Breast2}\}$, $\{\text{MouseLba}\}$, $\{\text{HumanLba}\}$, $\{\text{Necrosis}\}$, $\{\text{Lung}\}$, $\{\text{Glomeruli}\}$ and $\{\text{BoneMarrow}\}$ (see Table III). All other tasks were always incorporated however to train each multi-task model.

For the sake of completeness, we have also compared the transfer learning approaches with training from scratch and joint training. The former consists in training a network initialized with random weights. The latter consists in training a network in multi-task using the whole pool of tasks (including the target tasks).

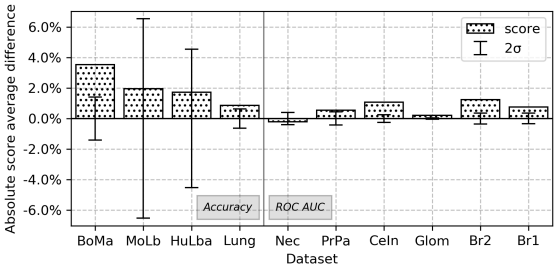
For training from scratch, we have used the same settings as for the fine-tuning experiment except for the network weights initialization (using initialization strategy as defined in PyTorch): same evaluated learning rates, number of training epochs and same networks. Regarding joint training, we have used the same architecture and training algorithm as for our multi-task pre-training. For the evaluation tasks listed above, only their training set is used for the multi-task training, while their validation and test sets were respectively used for optimizing the hyperparameters and evaluating the selected model. The data for the other training tasks were kept the same as for multi-task pre-training.

V. RESULTS AND DISCUSSION

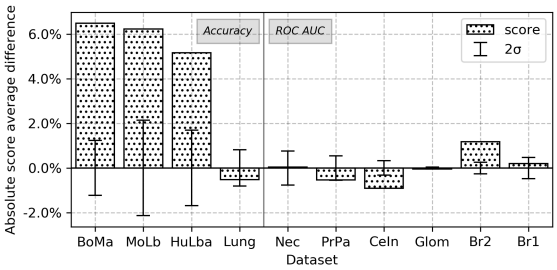
We report how our methods compare to transfer from ImageNet, training from scratch and joint training in Section V-A. Then, we study the effect on transfer of the various evaluated multi-task training hyperparameters in Section V-B. Finally, we discuss our results and future works in Section V-C.

A. Transfer performance

As explained in Section IV-D, we have used both feature extraction and fine-tuning to evaluate transfer performance. We could not repeat our experiments with several datasets splits given the computational cost, which would have allowed us to perform formal statistical tests for method comparison. To ease the discussions below, we will nevertheless call significant any difference between two average errors that exceed, in absolute value, twice the maximum of their standard deviations. If the scores were Gaussian distributed, this would ensure that each

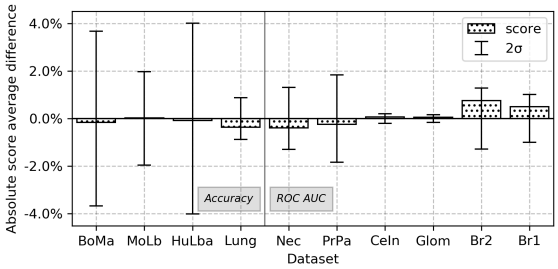


(a) DenseNet121

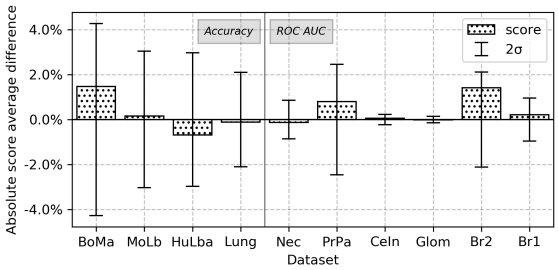


(b) ResNet50

Fig. 2. Absolute score difference between multi-task versus ImageNet pre-training using **feature extraction** as transfer protocol on our ten evaluation tasks. Positive difference indicates that multi-task pre-training yield superior performance. Tasks are sorted by evaluation metric and increasing dataset size. The variability of the multi-task transfer is measured using *two* standard deviations given by the error bars.



(a) DenseNet121



(b) ResNet50

Fig. 3. Absolute score difference between multi-task versus ImageNet pre-training using **fine-tuning** as transfer protocol on our ten evaluation tasks. See Figure 2 for details. Error bars are computed using two times the largest standard deviation among the ones resulting from ImageNet and multi-task fine tuning.

TABLE III
EVALUATION TASKS, THEIR EVALUATION METRIC, TRAINING SET SIZE AND FULL NAME.

Task	Train size	Full name	Metric
BoMa	652	BoneMarrow	*Accuracy*
MoLb	2438	MouseLba	
HuLba	4397	HumanLba	
Lung	5443	Lung	
Nec	791	Necrosis	*ROC AUC*
PrPa	1346	ProliferativePattern	
CeIn	1816	CellInclusion	
Glom	14605	Glomeruli	
Br2	14953	Breast1	
Br1	18261	Breast2	

average score is outside a 95%-confidence interval around the other score.

Absolute score differences between feature extraction from ImageNet and multi-task pre-trained networks can be found in Figures 2a and 2b for DenseNet121 and ResNet50 respectively. Our DenseNet121 features yield superior scores for nine datasets out of ten (all but *Necrose*) of which superiority is significant for all but two (*HumanLba* and *MouseLba*). The score difference is in favor of ImageNet features on *Necrose* although it is not significant. ResNet50 features yield superior results for six out of ten datasets (all but *CellInclusion*, *Glomeruli*, *ProliferativePattern* and *Lung*) of which superiority is significant for all but two datasets (*Necrose* and *Breast1*). Out of the four datasets where ImageNet features yield superior scores, the difference is significant for only one of them (*CellInclusion*). It is interesting to note that the largest difference of scores in favor of ImageNet is only 0.21% (ROC AUC) on *Necrose* for DenseNet121 whereas the difference in favor of our features is at most 3.52% (ACC) on *BoneMarrow*. Similary with ResNet50, the largest differences are 0.91% (ROC AUC) on *CellInclusion* and 6.48% (ACC) on *BoneMarrow* respectively in favor of ImageNet features and ours. Therefore, it appears that the loss of performance when our features underperform compared to ImageNet is lower than the expected gain of performance when our features are better. This indicates that the performance gain or loss you would obtain with multi-task features are dataset dependent and is hard to predict apriori, although the loss of performance is usually small compared to the possible improvement. Another interesting observation is the stability of transfer performance as only four evaluations (out of 20) exhibit standard deviations larger than 0.5% (ROC AUC or ACC).

Regarding fine-tuning, our features outperform ImageNet ones for five and six datasets with DenseNet121 and ResNet50 respectively (see Figures 3a and 3b). However, none of the differences are significant whether or not the advantage is in favor of our approach.

As a summary, feature extraction transfer approach seems to benefit from multi-task pre-training as 15 evaluations (out of 20) are in favor of our approach of which 11 are significant. Only two evaluations are significantly in favor of ImageNet features. However, fine-tuning from our features yield comparable performance with ImageNet features initialization as no score difference is significant (11 evaluations are in favor of our approach).

As an additional experiment, we compare transfer by feature extraction and fine-tuning with a similar model trained from

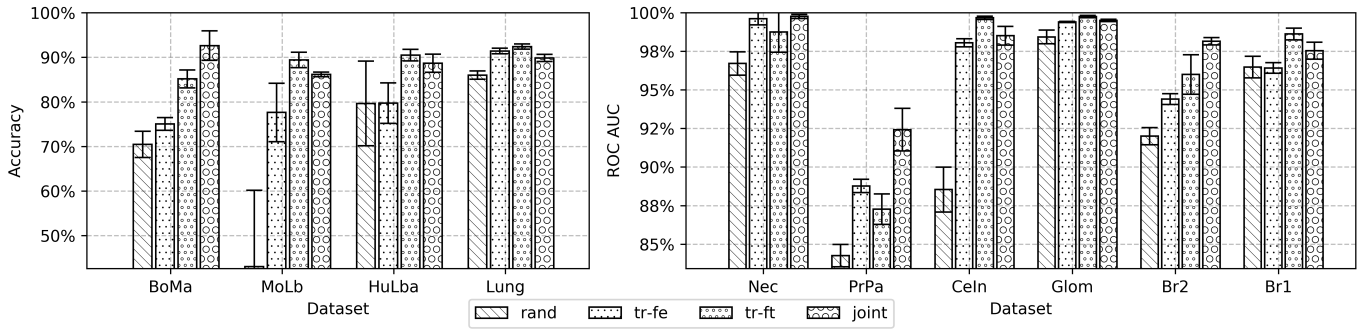


Fig. 4. Performance comparison of different approaches using DenseNet121: training from scratch (*rand*), feature extraction (*tr-fe*) and fine-tuning (*tr-ft*) using our multi-task pre-trained networks and joint training (*joint*). Each bar is the average performance and its error bar gives twice the standard deviation of a given method over five runs. For exact scores, see Supplementary Table V.

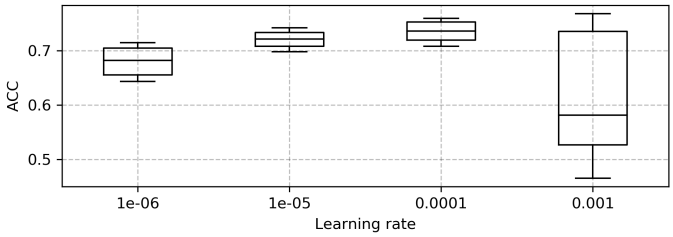
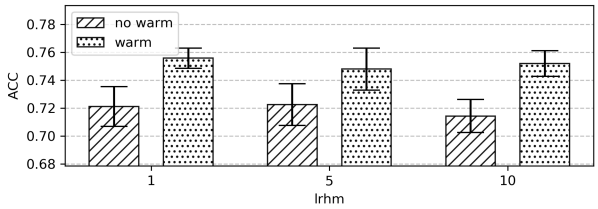
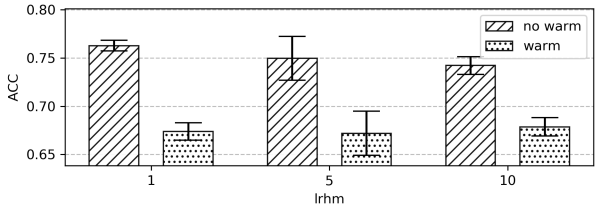


Fig. 5. Distributions of scores per learning rate on DenseNet121 with HumanLba dataset. Each boxplot results from the aggregation of the transfer scores of all models using the same learning rate value on the given network and dataset.



(a) HumanLba



(b) MouseLba

Fig. 6. Transfer performance for combinations of the hyperparameters γ_τ (learning rate heads multiplier, *lrhm*) and w (warm up) on two different datasets with learning rate $\gamma = 10^{-4}$ on DenseNet121. Error bars report twice the standard deviation.

scratch and the joint MTL approach described in Section IV-F (see Figure 4). These experiments show that fine-tuning improves over feature extraction significantly on most datasets and that training from scratch is subpar compared with the transfer approaches, feature extraction included. Both observations confirm previously published results [9], [27], [28], [30]. It appears that joint training significantly improves the performance on small datasets (*BoneMarrow* and *ProliferativePattern*) over all other approaches. For larger datasets, performance seem to lie between fine-tuning and feature extraction, or to be on par with fine-tuning on the datasets where the task is almost solved (ROC AUC or ACC close to 1).

B. Study of multi-task training hyperparameters

Our experiments have shown that the most impactful parameters on transfer performance was the multi-task training learning rate. Figure 5 shows the distributions of scores per learning rate using DenseNet121 and *HumanLba* dataset. We have picked this plot specifically because it exhibits the most frequent pattern regarding the effect of the learning rate. Similar plots for other datasets as well as ResNet50 can be found in Section D in the Supplementary Materials. We have observed that the highest learning rate 10^{-3} yields highly variable performance and is most of the times inferior to lower learning rates. It indicates that this specific value is too high to cope with the multi-task setting as it prevents the models from making use of the tasks information efficiently. This is likely due to training instabilities (convergence issues, noisy training loss) and overfitting. It appears that the lowest learning rate 10^{-6}, although yielding more stable performance, generally underperforms higher learning rates 10^{-5} and 10^{-4}. For both networks and most datasets, the latter learning rate 10^{-4} is the best performing on average.

The impact of the two other hyperparameters τ_γ and w seems to be minor for most datasets as variation stays within two standard deviations. Moreover, there is no pattern emerging from our experiments regarding those hyperparameters in general. Two exceptions to this observation are the *HumanLba* and *MouseLba* datasets which exhibit significant performance variations on both networks. The variations are shown in Figure 6 (similar figures can be found for other networks and datasets in Supplementary Section D). Those Figures show that *HumanLba* benefits from warming up whereas *MouseLba* performance are hurt. This indicates that the effect on transfer performance of warming up and multiplying heads learning rate is very dataset dependent and no general rule can be drawn

from our experiments.

C. Discussion and future works

Features extracted from our models are in general superior to ImageNet ones which shows that multi-task pre-training is effective at creating task-specific features. This important observation confirms the conclusions of previous works that domain-specific pre-training is a good idea and also validates the multi-task approach when a large source dataset is not available.

The second important observation is that fine-tuning does not benefit from using our models as we obtain similar transfer performance whatever the source. This indicates that fine-tuning is able to recover the lack of specificity of ImageNet features compared to ours. It contradicts our initial hypothesis that transfer should work better when source and target domains are similar. There might be several reasons why we observe such phenomenon. First, Yosinski *et al.* [18], who concluded about the importance of task similarity for efficient transfer, performed their experiments on simple architectures (*e.g.* simple stack of convolutional layers). In our case, we have used more recent architectures (*i.e.* residual and densely connected networks) which are easier to train and might be less impacted by their initial weights. Second, most previous works have shown that fine-tuning was beneficial in a single source task transfer scenario exclusively. Our multi-task pre-training is able to learn specific features but we can not exclude that a more advanced approach could result in even stronger features that might change our conclusion that fine-tuning does not benefit from MTL transfer. For instance, some papers have highlighted training difficulties associated with multi-task (*e.g.* gradients interference [50]) and transfer learning (*e.g.* batch normalization [45]) that could be investigated in our context. In addition, it is likely that, given a target task, not all tasks in the pool contribute equally to transfer performance. Some tasks might even have a destructive effect during the pre-training phase (*i.e.* if they were removed, transfer performance would increase). Incorporating a training mechanism that could dynamically increase (*resp.* decrease) the contribution of constructive (*resp.* destructive) tasks would certainly help improving the resulting features. Alternatively, instead of using all the tasks, one could find a mechanism for selecting a subset of (the most relevant) source tasks for a given target task. Such solution would entail however a significant additional computational cost, since a new MTL model would have to be trained for each new target task.

There are also several interesting research directions regarding the architecture. On the one hand, it would be interesting to study the effect of increasing the capacity of the task-specific parts. On the other hand, we have only worked with classification tasks so far but it is possible to incorporate directly segmentation or detection tasks to the pre-training by appending an ad-hoc network as a head. Hopefully, enriching training signal with a dense prediction task such as segmentation could improve the transferability of the resulting models. However, this approach also raises pratical questions and issues such as model memory usage or loss aggregation.

More practically, we plan to integrate newly released datasets into our pool to reinforce its versatility and hopefully the transferability of the pre-trained models. We also plan to integrate the pre-trained models to the Cytomine open-source tool [39] and the BIAFLOWS benchmarking platform [51].

VI. Conclusion

In this work, we have investigated the use of multi-task learning for pre-training neural networks in digital pathology. We have first created a pool of classification tasks from existing sources containing almost 900k digital pathology images. Using this pool, we have pre-trained a neural network in a multi-task setting in order to transfer the resulting model to unseen digital pathology tasks. Using a robust evaluation protocol, we have shown that transferring a model pre-trained in multi-task can be beneficial for the performance on the target task. When compared to transfer from ImageNet, our pre-training approach coupled with feature extraction yields comparable or better performance depending on the target dataset. We have observed that fine-tuning multi-task or ImageNet pre-trained models yields comparable performance. It suggests that fine-tuning is able to recover from the lack of feature specificity whatever the pre-training source. However, pre-training remains crucial, as models trained from scratch are clearly inferior.

Acknowledgments

The authors would like to thank all previous authors and scientists who released their datasets (see Supplementary Section F) as well as Joeri Hermans and Ulysse Rubens for technical support. RaM was supported by IDEES grant with the help of the Wallonia and the European Regional Development Fund (ERDF). Computational infrastructure is partially supported by ULiège, Wallonia, and Belspo funds.

References

[1] L. Roux, D. Racoceanu, F. Capron, J. Calvo, E. Attieh, G. Le Naour, and A. Gloaguen, "Mitos & atypia," *Image Pervasive Access Lab (IPAL), Agency Sci., Technol. & Res. Inst. Infocom Res., Singapore, Tech. Rep*, vol. 1, pp. 1–8, 2014.

[2] K. Sirinukunwattana, S. e Ahmed Raza, Y.-W. Tsang, D. R. Snead, I. A. Cree, and N. M. Rajpoot, "Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images." *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1196–1206, 2016.

[3] A. Janowczyk and A. Madabhushi, "Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases," *Journal of pathology informatics*, vol. 7, 2016.

[4] N. Linder, J. Konsti, R. Turkki, E. Rahtu, M. Lundin, S. Nordling, C. Haglund, T. Ahonen, M. Pietikäinen, and J. Lundin, "Identification of tumor epithelium and stroma in tissue microarrays using texture analysis," *Diagnostic pathology*, vol. 7, no. 1, p. 22, 2012.

[5] M. Veta, Y. J. Heng, N. Stathonikos, B. E. Bejnordi, F. Beca, T. Wollmann, K. Rohr, M. A. Shah, D. Wang, M. Rousson *et al.*, "Predicting breast tumor proliferation from whole-slide images: the tupac16 challenge," *Medical image analysis*, vol. 54, pp. 111–121, 2019.

[6] G. Aresta, T. Araújo, S. Kwok, S. S. Chennamsetty, M. Safwan, V. Alex, B. Marami, M. Prastawa, M. Chan, M. Donovan *et al.*, "Bach: Grand challenge on breast cancer histology images," *Medical image analysis*, 2019.

[7] B. E. Bejnordi, M. Veta, P. J. Van Diest, B. Van Ginneken, N. Karssemeijer, G. Litjens, J. A. Van Der Laak, M. Hermsen, Q. F. Manson, M. Balkenhol *et al.*, "Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer," *Jama*, vol. 318, no. 22, pp. 2199–2210, 2017.

[8] J. N. Kather, C.-A. Weis, F. Bianconi, S. M. Melchers, L. R. Schad, T. Gaiser, A. Marx, and F. G. Zöllner, "Multi-class texture analysis in colorectal cancer histology," *Scientific reports*, vol. 6, p. 27988, 2016.

[9] R. Mormont, P. Geurts, and R. Marée, "Comparison of deep transfer learning strategies for digital pathology," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2262–2271.

[10] R. Marée, S. Dallongeville, J.-C. Olivo-Marin, and V. Meas-Yedid, "An approach for detection of glomeruli in multisite digital pathology," in *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*. IEEE, 2016, pp. 1033–1036.

[11] P. Kainz, H. Burgsteiner, M. Asslaber, and H. Ahammer, "Training echo state networks for rotation-invariant bone marrow cell classification," *Neural Computing and Applications*, vol. 28, no. 6, pp. 1277–1292, 2017.

[12] H. R. Tizhoosh and L. Pantanowitz, "Artificial intelligence and digital pathology: Challenges and opportunities," *Journal of pathology informatics*, vol. 9, 2018.

[13] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017.

[14] S. Robertson, H. Azizpour, K. Smith, and J. Hartman, "Digital image analysis in breast pathology—from image processing techniques to artificial intelligence," *Translational Research*, vol. 194, pp. 19–35, 2018.

[15] D. Komura and S. Ishikawa, "Machine learning methods for histopathological image analysis," *Computational and structural biotechnology journal*, vol. 16, pp. 34–42, 2018.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[17] H. Shang, Z. Sun, X. Fu, Z. Zhang, and W. Yang, "What and how other datasets can be leveraged for medical imaging classification," in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE, 2019, pp. 814–818.

[18] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[19] U. A. H. Khan, C. Stürenberg, O. Gencoglu, K. Sandeman, T. Heikkinen, A. Rannikko, and T. Mirtti, "Improving prostate cancer detection with breast histopathology images," *arXiv preprint arXiv:1903.05769*, 2019.

[20] A. Medela, A. Picon, C. L. Saratxaga, O. Belar, V. Cabezón, R. Cicchi, R. Bilbao, and B. Glover, "Few shot learning in histopathological images: Reducing the need of labeled data on biological datasets," in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE, 2019, pp. 1860–1864.

[21] O. Z. Kraus, B. T. Grys, J. Ba, Y. Chong, B. J. Frey, C. Boone, and B. J. Andrews, "Automated analysis of high-content microscopy data with deep learning," *Molecular systems biology*, vol. 13, no. 4, p. 924, 2017.

[22] Y. Zhang and Q. Yang, "A survey on multi-task learning," *arXiv preprint arXiv:1707.08114*, 2017.

[23] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[24] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[25] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. IEEE, 2014, pp. 512–519.

[26] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?" in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 2661–2671.

[27] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[28] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[29] M. Babaie and H. R. Tizhoosh, "Deep features for tissue-fold detection in histopathology images," in *Digital Pathology*, C. C. Reyes-Aldasoro, A. Janowczyk, M. Veta, P. Bankhead, and K. Sirinukunwattana, Eds. Cham: Springer International Publishing, 2019, pp. 125–132.

[30] F. Ponzio, G. Urgese, E. Ficarra, and S. Di Cataldo, "Dealing with lack of training data for convolutional neural networks: The case of digital pathology," *Electronics*, vol. 8, no. 3, p. 256, 2019.

[31] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, vol. 4, 2017, p. 12.

[32] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[33] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3994–4003.

[34] G. Strezoski, N. van Noord, and M. Worring, "Many task learning with task routing," *arXiv preprint arXiv:1903.12117*, 2019.

[35] R. K. Samala, H.-P. Chan, L. M. Hadjiiski, M. A. Helvie, K. H. Cha, and C. D. Richter, "Multi-task transfer learning deep convolutional neural network: application to computer-aided diagnosis of breast cancer on mammograms," *Physics in Medicine & Biology*, vol. 62, no. 23, p. 8894, 2017.

[36] W. Zhang, R. Li, T. Zeng, Q. Sun, S. Kumar, J. Ye, and S. Ji, "Deep model based transfer and multi-task learning for biological image analysis," *IEEE transactions on Big Data*, 2016.

[37] X. Pan, L. Li, H. Yang, Z. Liu, Y. He, Z. Li, Y. Fan, Z. Cao, and L. Zhang, "Multi-task deep learning for fine-grained classification/grading in breast cancer histopathological images," in *International Symposium on Artificial Intelligence and Robotics*. Springer, 2018, pp. 85–95.

[38] E. Arvaniti and M. Claassen, "Coupling weak and strong supervision for classification of prostate cancer histopathology images," *arXiv preprint arXiv:1811.07013*, 2018.

[39] R. Marée, L. Rollus, B. Stévens, R. Hoyoux, G. Louppe, R. Vandaele, J.-M. Begon, P. Kainz, P. Geurts, and L. Wehenkel, "Collaborative analysis of multi-gigapixel imaging data using cytomine," *Bioinformatics*, vol. 32, no. 9, pp. 1395–1401, 2016.

[40] M. Babaie, S. Kalra, A. Sriram, C. Mitcheltree, S. Zhu, A. Khatami, S. Rahnamayan, and H. R. Tizhoosh, "Classification and retrieval of digital pathology scans: A new dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 8–16.

[41] J. Gamper, N. A. Koohbanani, K. Benet, A. Khuram, and N. Rajpoot, "Pannuke: An open pan-cancer histology dataset for nuclei instance segmentation and classification," in *European Congress on Digital Pathology*. Springer, 2019, pp. 11–19.

[42] F. A. Spanhol, L. S. Oliveira, C. Petitjean, and L. Heutte, "A dataset for breast cancer histopathological image classification," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 7, pp. 1455–1462, 2015.

[43] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[44] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," *Pattern Recognition*, vol. 80, pp. 109–117, 2018.

[45] W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han, "Domain-specific batch normalization for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7354–7362.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[47] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, no. 2, 2017, p. 3.

[48] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[49] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[50] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *arXiv preprint arXiv:2001.06782*, 2020.

[51] U. Rubens, R. Mormont, V. Baecker, G. Michiels, L. Paavolainen, G. Ball, D. Ünay, B. Pavie, A. Chessel, L. A. Scholz *et al.*, "Biaflows:

A collaborative framework to benchmark bioimage analysis workflows,"
To appear in Cell Patterns, 2020.