Interpreting mechanisms of prediction for skin cancer diagnosis using multi-task learning

Davide Coppola
Bioinformatics Institute
A*STAR, Singapore
davidec@bii.a-star.edu.sg

Hwee Kuan Lee
Bioinformatics Institute
A*STAR, Singapore
leehk@bii.a-star.edu.sg

Cuntai Guan
SCSE
NTU, Singapore
CTGuan@ntu.edu.sg

Abstract

One of the key issues in deep learning is the difficulty in the interpretation of mechanisms for the final predictions. Hence the real-world application of deep learning in skin cancer still proves limited, in spite of the solid performances achieved. We present a way to better interpret predictions on a skin lesion dataset by the use of a multi-task learning framework and a set of learnable gates. The model detects a set of clinically significant attributes in addition to the final diagnosis and learns the association between tasks by selecting which features to share among them. Conventional multi-task learning algorithms generally share all the features among tasks and lack a way of determining the amount of sharing between tasks. On the other hand, this method provides a simple way to inspect which features are being shared between tasks in the form of gates that can be learned in an end-to-end fashion. Experiments have been carried out on the publicly available Derm7pt dataset, which provides diagnosis information as well as the attributes needed for the well-known 7-point checklist method.

1. Introduction

Skin cancer is one of the most common forms of cancer, for which an early diagnosis has been shown to reverse the odds in the majority of cases [16]. Deep learning methods for the diagnosis of skin cancer have already reached dermatologist-level performance [8, 14, 16, 34]. However, there appears to be a discrepancy between the performance and the utilisation in real clinical settings of these models, mainly due to the barriers in understanding the choices done by deep learning algorithms. Thus, one of the next impactful directions in this area of research should be the interpretation of the internal mechanisms of a model for the final diagnosis. As a matter of fact, the interpretation of the behaviour in deep learning is a key missing component in making this technology applicable and trustworthy in many real-world scenarios. In medical applications, where the well-being of humans is a concern, there is a particular need for models that can show their mechanism in an easily interpretable way for physicians. It is not surprising that this particular branch of research has been recently getting increased traction under the name of eXplainable AI (XAI) [6]. One way of approaching this problem is to "look into" the hidden activations of neural networks to explain the prediction mechanism [22, 28]. This is akin to neuroscience, where researchers look into the activity of neurons in the brain to understand its functioning: a rather difficult task given the high complexity of our brain, just like a deep learning model. In fact, in spite of their promising performance, most state-of-the-art deep learning models are made up of thousands or millions of parameters that do not have a clear meaning for us. However, humans do not explain themselves through the activation of their neurons. Explanations are instead usually based on associations between elements, logic and causal reasoning. Indeed, by association alone, we could interpret our predictions fairly well. For this reason, we have worked on the interpretation of deep neural networks for skin cancer diagnosis using a multi-task learning (MTL) [10] framework, as the associations between tasks related to the analysis of a skin lesion can help with understanding the behaviour of the model. To pursue this, we devised a strategy based on gate elements, that allows us to directly interpret the amount of sharing of information between the different tasks. Historically, two rule-based algorithms, namely ABCD rule [24] and 7-point checklist [4], have been used by doctors for the detection of melanoma. Both are based on the detection of a series of attributes and patterns in the lesion. Each has its own strengths and weaknesses, with the latter proving to have a higher sensitivity, in exchange for a lower specificity [9]. The fact that rule-based methods have been widely adopted by clinicians proves their merit. Thus, the analysis of skin images through a rule-based method can be inherently framed as a MTL problem and provide a use-case for

the developed model. Experiments have been carried out on the Derm7pt dataset [19], which provides the diagnosis supported by the clinically significant attributes that are part of the 7-point checklist method.

2. Related works

2.1. On Multi-Task Learning

Multi-Task Learning (MTL) [10] is a branch of Machine Learning in which an algorithm is developed with the aim of learning multiple tasks at the same time, usually from a single set of data. It is motivated by the idea that related tasks might share a representation of the data and information learned for one task might be useful for another one. Conceptually, this is a process easily handled by humans, who are naturally able to find correlations between tasks and exploit them to improve their overall performance. Similarly, having multiple objective functions introduces an inductive bias to the model, which causes it to lean towards configurations that are optimal for more than one task [26]. The overall effect is that MTL increases the generalization capabilities of the model [26] and the data efficiency, as it can be seen as a form of implicit data augmentation [26]. Thus, learning multiple tasks at same time seems an efficient solution for those cases where the number of available data points might not be sufficient to optimize millions of parameters [37]. However, just increasing the number of tasks learned together is not guaranteed to provide all these benefits as the intra-task relatedness plays a fundamental role, to the point that it might even have a disruptive effect if vaguely related tasks start competing for the parameter space [29, 30].
With regards to supervised deep learning, MTL architectures can be mostly divided in two main categories [26]: *hard parameter sharing*, in which the model has a fixed architecture that is shared between the tasks, and *soft parameter sharing*, where usually each task has its own model and the sharing occurs through means of constraints to the task-specific model parameters or through their combination according to a defined rule. *Cross-stitch networks* [23] model the shared knowledge between tasks through a linear combination of their feature maps after each convolutional layer, carried out through a combinatorial unit (namely *cross-stitch unit*) whose weights are learned during training. The concept is further generalized in [27], which introduces the MTL meta-architecture known as *Sluice Networks*. The parameter space of each convolutional layer is split in two subspaces: task-specific and shared. Their outputs are mixed through a linear combination mediated by parameters learned during training. Furthermore, before the last layers, another linear combination is evaluated by combination of *skip* connections coming from previous layers. Given its design, this meta-architecture also generalizes other known MTL architectures, such as [10] and [23].
In *Task Routing* [31] a task-specific mask is defined for each convolutional layer in a dropout-like fashion, defining whether a feature map is used by one task or the other. A sharing ratio hyper-parameter defines the amount of features maps that are shared between tasks. As this method only introduces a mask layer, it can be applied to any pre-existing architecture with relative ease of implementation.
The *Neural Discriminative Dimensionality Reduction (NDDR) CNN* architecture [15] models the question of learning what to share as a dimensionality reduction problem using known CNN components. The *NDDR* block is introduced, a combination of a 1×1 convolution and batch normalization, on which weight decay is applied for regularization, that is inserted after the feature maps from each task are concatenated together. As a matter of fact, a 1×1 convolution is equivalent to a discriminative dimensionality reduction, as one parameter per channel is learned that acts as coefficient in the linear combination of channels. Similarly to [31], this solution can be applied to pre-existing fixed architectures with little parameter overhead as a "plug and play" kind of solution.

2.2. On MTL and Skin Lesions

While most works in the context of skin lesions focus on the single tasks of directly classifying a lesion or segmenting it [7, 8, 13, 14, 16, 17, 18, 33], some try to use the benefits of MTL to develop unified pipelines dealing with multiple tasks [3, 11, 12, 19, 35, 36]. This work is similar to [19], which proposes a Multi-Modal MTL framework to classify skin lesions from the [5] dataset, which includes pictures captured with diverse modalities, metadata and the annotations related to the 7-point checklist. The authors use all the data at their disposal to develop an architecture that can predict the diagnosis of the lesion and the 7-point checklist attributes in a single optimization, guided by a weighed multi-modal multi-task loss function. Given that the dataset is highly unbalanced, they propose a sampling strategy during training such that at every iteration every label gets a similar representation and each sample's contribution to the loss function is weighed according to its label's representation in the mini-batch. The impact of the different modalities on the output is investigated by the authors, as well as a comparison between the direct diagnosis of melanoma (i.e. output of the network) and the diagnosis inferred from the predicted 7-point checklist attributes. The same dataset from [19] has been used for experiments in [3], which introduces a MTL architecture using CNNs and makes use of both the dermoscopic and clinical images. The proposed model consists of one CNN for each of the checklist attributes, where the CNN has two independent branches (one for the dermoscopic and one for the clinical image input) that merge after the feature maps are flattened. While the

t	task name	labels
0	Diagnosis (DIAG)	BCC, NEV, MEL, MISC, SK
1	Pigment Network (PN)	ABS, TYP, ATP
2	Blue Whitish Veil (BWV)	ABS, PRS
3	Vascular Structures (VS)	ABS, REG, IR
4	Pigmentation (PIG)	ABS, REG, IR
5	Streaks (STR)	ABS, REG, IR
6	Dots and Globules (DaG)	ABS, REG, IR
7	Regression Structures (RS)	ABS, PRS

Table 1: Summary of the 8 tasks learned by the algorithm. The first task (DIAG) indicates the final diagnosis of the lesion, whereas the following are the 7 attributes that are used in the 7-point checklist. *BCC*: basal cell carcinoma; *NEV*: nevus; *MEL*: melanoma; *MISC*: miscellaneous; *SK*: seborrheic keratosis; *ABS*: absent; *TYP*: typical; *ATP*: atypical; *PRS*: present; *REG*: regular; *IR*: irregular.

architecture for each block is identical, the weights are not shared. The outputs from the network are then mapped to the 7-point checklist scores and, following the classic rules used by doctors, the final diagnosis prediction is given. As this method uses the 7-point checklist, it is only able to discern melanoma and non-melanoma cases; by contrast, [19] also has a direct diagnosis output, which allows a finer grain classification of the lesions in 5 classes.

3. Dataset and the 7-point checklist

The 7-point checklist [4] is a rule-based method used by dermatologists to evaluate whether the skin lesion being analyzed is a melanoma or not. The method consists in looking for 7 specific patterns on the lesion (Table 1) and evaluating their status; i.e. a pattern may be absent or present and showing a typical or atypical conformation. Depending on the evaluation, each pattern is attributed a score. If the sum of the scores exceeds a clinically established threshold τ, the lesion is diagnosed as melanoma. Prominent values for the threshold are $\tau = 1$ and $= 3$, with the former providing higher sensitivity to melanoma in spite of lower overall precision.

The data used in this work is the Derm7pt dataset (from the *Interactive Atlas of Dermoscopy* [5]), which was publicly released with [19]. The dataset consists of 1011 cases of skin lesions annotated by experts. For each case, data are available in multiple modes (e.g. metadata, images), however this work only considers the dermoscopic images as data source, as they provide a higher resolution and allow to appreciate better the patterns on the lesion that are necessary for the criteria. Labels are available for 8 categories with varying degrees of granularity as summarized in Table 1; these categories will be used as the 8 tasks to learn in the MTL architecture. Thus, in the following the terms *category* and *task* will be used interchangeably.
The pre-processing procedure of the dataset, consisting in grouping the more infrequent labels and splitting in train-

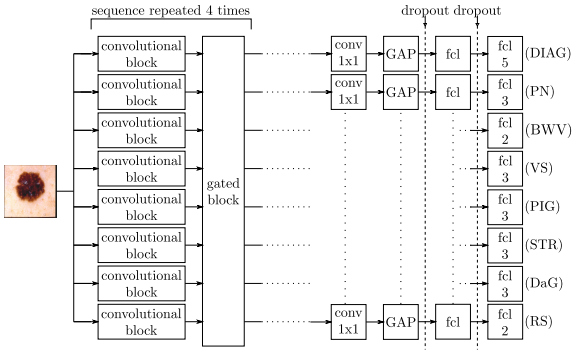


Figure 1: Representation of the complete model architecture. The sequence of convolutional block and gated block is repeated 4 times before the final sequence of layers. *GAP*: global average pooling; *fcl*: fully connected layer, where the number of output neurons is indicated.

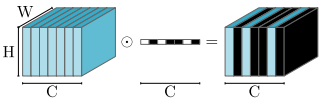


Figure 2: Representation of how a gate works. A tensor of spatial dimensions $W \times H$ and C feature maps is multiplied over the channel axis by the gate, a vector of values that learns which features to suppress (0, black) or select (1, white). Note that the the values of the gates are continuous in the interval $[0, 1]$.

validation-test sets, has been executed just like in [19][1]. The number of samples per set are as follows: 413 cases for training, 203 cases for validation and 395 cases for the testing phase.

4. Methods

The aim of this work is to develop a single neural network that, given one input dermoscopic image, is able to classify the diagnosis of the skin lesion, as well as seven clinically significant patterns that make up the 7-point checklist (Section 3). The whole framework is represented in Figure 1.

It is desirable to develop a MTL method that learns what to share between tasks [15, 31]. The idea is that different tasks should only receive information that is relevant to them from other tasks, thus it is necessary to develop a way to learn how to discriminate among features passed from other tasks. We use gates to select which features should be kept (i.e. multiply by 1) or suppressed (i.e. multiply by 0) as shown in Figure 2. While the suppressed features are not really removed from the feature tensor, being multiplied by 0 makes them virtually null in the following con-

[1] Code available at https://github.com/jeremykawahara/derm7pt.

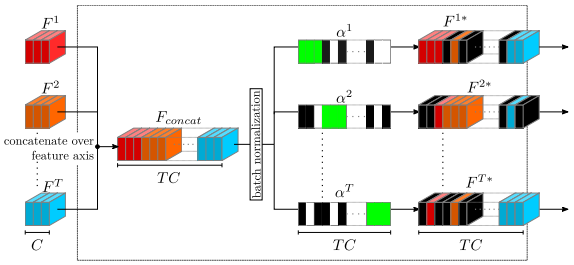


Figure 3: Representation of a *gated block* for T tasks, where each input F^t has $C = 3$ channels. The feature maps are concatenated along the channel axis and passed through the task specific gates. The colors *white* and *black* indicate an open ($\alpha^t_{i,c} = 1$) or closed ($\alpha^t_{i,c} = 0$) gate, respectively; the color *green* indicates constant values, since the features from the same task are not filtered by the gates ($\alpha^t_{t,c} = 1$, constant).

volutional layer. In order to learn the values of the gate through standard optimization techniques in an end to end fashion and make backpropagation possible, it is necessary to model them as a vector of continuous values over which an activation function is applied to map them in the interval $[0, 1]$ (Section 4.1).

Formally, a set is given of n color images $x_s \in \mathcal{X}$ and their corresponding one-hot encoded labels for T tasks

$$\mathbf{y}_s = \{\mathbf{y}^1_s, \mathbf{y}^2_s, \dots \mathbf{y}^T_s\} \in \mathcal{Y} \subset \mathbb{R}^{J^1} \otimes \mathbb{R}^{J^2} \otimes \cdots \mathbb{R}^{J^T} \quad (1)$$

where $\otimes$ is the direct product, $s = 1, \dots n$ is the sample index and $J^1, \dots J^T$ indicate the number of classes in each task t. Thus, each

$$\mathbf{y}^t_s = [y^t_{s,1},\ y^t_{s,2} \dots y^t_{s,J^t}] \in \mathbb{R}^{J^t} \quad (2)$$

is the vector of one-hot encoded labels of sample s for task t. The objective of the MTL problem is to find a neural network g_Θ, with a set of parameters Θ, such that $g_\Theta : \mathcal{X} \mapsto \mathcal{Y}$.

4.1. Gated block

The idea proposed in this work is to have learnable "gates" that determine which features coming from other tasks would be useful for the specific task they refer to. A similar concept has been implemented, with different objectives, also in [2, 15, 32].

Consider having T tasks, each one carried out by a task-specific neural network with the same architecture. A *gated block* can be applied after any convolutional block, to share features with the same spatial resolution; one gate will be initialized for each task. A convolutional block can be any sequence of CNN operations (e.g. convolution, max-pooling, residual block). Since the feature maps will be concatenated, it is necessary for all the task-specific convolution blocks to output features maps with the same spatial resolution. For simplicity, all the task-specific convolutional blocks are chosen to have the have the same number of output feature maps.

A *gated block* (represented in Figure 3) concatenates the features coming from the T task-specific convolutional blocks on the channel axis, applies batch normalization to the new set of features and then passes them through a set of *gates* α^t that determine which features should be shared. Each gated block consists of T independent gates, one for each task, with the aim to select which features to share. The gate α^t for task t should always keep the features coming from task t, while selecting or rejecting those coming from other tasks.

A gated block is formally defined as an operational unit that takes a set of feature maps $F = \{F^1, F^2, \dots, F^T\}$ as input and outputs a set of processed feature maps $F^* = \{F^{1*}, F^{2*}, \dots, F^{T*}\}$. The tensor F^t indicates the input coming from a convolutional block of task t; similarly F^{t*} will be the output of the gated block concerning task t. Each of these can also be written unwrapping the channel dimension as

$$F^t = [F^t_1, F^t_2, \dots, F^t_C] \qquad \in \mathbb{R}^{H \times W \times C} \quad (3)$$

$$F^{t*} = [F^{t*}_1, F^{t*}_2, \dots, F^{t*}_{TC}] \qquad \in \mathbb{R}^{H \times W \times TC} \quad (4)$$

where each F^t_c or F^{t*}_c represents the c-th feature map of the tensor regarding task t.

The input feature maps are concatenated along the feature axis, yielding

$$F_{concat} = \left[F^1, F^2 \dots F^T\right] = \quad (5)$$
$$= \left[F^1_1, F^1_2, \dots F^1_1, F^2_2, \dots F^T_{C-1}, F^T_C\right] \in \mathbb{R}^{H \times W \times TC}$$

This is then multiplied by each of the T task-specific gates to obtain the tensor $F^{t*} = F_{concat}\ \alpha^t$.

Ideally a gate would be a made up of binary values (either 0 or 1), that would indicate whether that feature should be shared between tasks; thus functioning like a mask. However, to be able to learn the gates through gradient descent in an end-to-end fashion it is necessary to model them as a vector of continuous values

$$\hat{\alpha}^t = \left[\hat{\alpha}^t_{1,1}, \hat{\alpha}^t_{1,2}, \dots, \hat{\alpha}^t_{i,c}, \dots, \hat{\alpha}^t_{T,C}\right]^{\mathrm{T}} \in R^{TC} \quad (6)$$

where each $\hat{\alpha}^t_{i,c} \in \mathbb{R}$ refers to the "raw" value of the coefficient for the c-th feature map coming from task i in the gate of task t.

The raw values of the vector are subsequently mapped in the interval $[0, 1]$ through means of the sigmoid function, as follows

$$\alpha^t_{i,c} = \frac{1}{1 + e^{-\gamma_g \hat{\alpha}^t_{i,c}}} \in [0, 1] \quad (7)$$

where γ_g is a hyperparameter defining the shape of the curve to make it more similar to a step function. A generic $\alpha^t_{i,c}$ indicates the value that will multiply the c-th feature map coming from task i in the gate of task t.
Furthermore, we want the gates to be applied only to the features coming from other tasks, while the features specific to that task should always be left intact; to achieve this, the gates are initialized as $\hat{\alpha}^t_{i,c} = 100$ (non-trainable) if $t = i$ and 0 otherwise. With this initialization, all the gates start with the same constant value, meaning that the contribution of each feature map coming from other tasks will be equal. Modelled in this way, the algorithm can learn the "raw" values $\hat{\alpha}^t_{i,c}$ through standard optimization techniques.

Similarly to NDDR-CNN [15], features among tasks are shared through concatenation on the feature axis followed by an operation to reduce the dimensionality. In [15] this is achieved directly through the sequence of 1×1 convolution followed by batch normalization for each task, plus L_2 regularization on these kernels. This work differs in the introduction of one "gate" element for each task after the concatenation, as described above. The gates model the desired behaviour of selecting the useful features from another task while suppressing the others. A 1×1 convolution to reduce the feature channel dimension is also applied here, after each gate. Furthermore, batch normalization is applied after the all the feature maps are concatenated, instead of after the dimensionality reduction. Ideally, a 1×1 filter such as the one in NDDR-CNN [15] could learn the same behaviour expressed by the proposed gates. However, including them explicitly after the concatenation allows to enforce the desired behaviour of selectivity in the model.

4.2. Training matters

Balanced mini-batches Due to the imbalance in the dataset and the number of tasks, it may happen that multiple mini-batches do not include one of the unique labels, meaning that the model will rarely be optimized for that label. Thus, this work uses the same mini-batch sampling strategy described in [19].
At each training iteration, k elements are randomly sampled from the training set for each of the unique labels present in the dataset, so that every mini-batch consists of at least k elements belonging to each unique label. As noted in Table 1, there are 24 unique labels in the dataset across the 8 tasks considered; thus the batch size will be $b = 24k$. Since the unique labels are not mutually exclusive, some of them may repeat more than k times in a mini-batch, which is why [19] makes use of dynamic weights, computed at each training iteration for each unique label. For each task, a vector of weights $\mathbf{w}^t = [w^t_1, \dots, w^t_{J^t}] \in \mathbb{R}^{J^t}$ is computed in a way that gives a higher value to the more infrequent labels for task t in the current mini-batch. This is achieved through the equation

$$\mathbf{w}^t(\mathbf{y}^t) : w^t_j = \frac{\max\left(\mathbf{1}\,\mathbf{y}^t\right)}{\left(\mathbf{1}\,\mathbf{y}^t\right)_j} \tag{8}$$

detailed in [19]; where according to Eq. 2 $\mathbf{y}^t = [\mathbf{y}^t_1, \mathbf{y}^t_2 \dots \mathbf{y}^t_b]^{\mathrm{T}} \in \mathbb{R}^{b \times J^t}$ is the tensor of one-hot encoded labels for the b samples in the mini-batch. As $k > 0$, there will always be at least one sample in the mini-batch belonging to each unique label, avoiding the possibility of having a division by zero in Eq. 8.

Loss function All the tasks are classification tasks, which can be associated with a *focal cross-entropy loss* as defined in [21]. The *focal* cross-entropy loss scales the standard cross-entropy by a factor $(1 - \tilde{y}^t_{s,j})^\beta$, which indicates how hard the sample was to classify; i.e. samples that are easily classified by the network have higher *confidence* and should have a lower impact on the loss function. β is a hyperparameter set to $\beta = 2$ according to the findings in [21].
Since each sample's contribution will be weighted based on the number of elements with that label in the mini-batch (as detailed in the previous section), the classification loss function for each task for one sample s is

$$\mathrm{FL}(\mathbf{y}^t_s, \tilde{\mathbf{y}}^t_s, \mathbf{w}^t(\mathbf{y}^t)) = -\sum_j^{J^t} w^t_j \, y^t_{s,j} \, (1 - \tilde{y}^t_{s,j})^\beta \, \log(\tilde{y}^t_{s,j}) \tag{9}$$

where $\mathbf{y}^t_s \in \mathbb{R}^{1 \times J^t}$ and $\tilde{\mathbf{y}}^{\mathbf{t}}_{\mathbf{s}} \in \mathbb{R}^{1 \times J^t}$ are the one-hot encoded ground truth and the softmax-activated predicted output of the network, respectively as defined in Eq. 1, 2. $\mathbf{w}^t(\mathbf{y}^t)$ is the vector of mini-batch weights for task t, computed according to Eq. 8.
Considering all the tasks and the regularization penalties, the final loss function is

$$\mathcal{L} = \frac{1}{b} \sum_t^T \sum_s^b \mathrm{FL}(\mathbf{y}^t_s, \tilde{\mathbf{y}}^t_s, w(\mathbf{y}^t)) \quad + \gamma_{L_2} \sum_{\forall \theta \in \Theta_K} \frac{||\theta^2||}{2} \tag{10}$$

where the last term represents the L_2 regularization applied to all the kernels in the model, indicated by the set $\Theta_K \subset \Theta$.

5. Experiments

The network used in the experiments is shown in Figure 1: the sequence of *convolutional block* and *gated block* is repeated 4 times before the final sequence of layers. The outputs of the last gated block go through task-specific (i.e. non-shared weights) blocks composed of a 1×1 convolution, global average pooling, and two fully connected layers, the last of which presents a softmax activation for the

final prediction. A convolutional block is defined as a sequence of a convolutional layer and max-pooling. For increased regularization, batch normalization is applied on the tensor of concatenated features in every gated block, with a momentum of 0.90. Furthermore, dropout with rate 0.3 is applied before the fully connected layers toward the end of the network. Additional details on the network configuration can be found in Table S1 in the supplementary material. The parameters in the algorithm are updated using Adam optimizer [20] with learning rate $\mu = 1\text{e}-3$ for the network parameters and $\mu_\alpha = 100\ \mu$ for the values of the gates. After testing different values, we chose $\gamma_g = 3$ (Eq. 7) as it allows us to have the desired behaviour on the gates. The regularization coefficient for L_2 has value $\gamma_{L_2} = 1\text{e}-4$. The sampling strategy described in Section 4.2 is always used during training in order to have at least $k = 6$ samples of each class per mini-batch. Since these are randomly sampled from the training dataset with reinsertion, an epoch is not defined as the iteration over all the samples in the set. We define an epoch as a fixed number of 5 training iterations. Early stopping has been implemented with a patience of 50 epochs, monitoring the value of the validation loss. During validation and testing, the mini-batch weights (Eq. 8) are not computed, because it would not make sense to evaluate them at inference time given that they are highly dependent on the batch size. Similarly the L_2 penalty is not computed during inference, as it would just provide constant values. In the experiments (Sec. 5, the images have been resized to 256×256 pixels and random transformations (e.g. rotation, flipping, shift, shear) are applied during training as a data augmentation strategy.

Results are reported for the following experiments:

a) `standard`: the main architecture used in the experiments; the other experiments are built as variations of this architecture. Given the limited number of samples, the architecture has a small number of filters per each convolution in order to avoid overfitting.

b) `binary`: in this experiment the labels for the diagnosis task (DIAG) have been grouped in two categories in order to have a binary classification of "melanoma" vs "all". The architecture is identical to `standard`.

c) `half`: this experiment tests the effect on the model when *halving* the number of available parameters with respect to `standard`; i.e. for each convolutional layer, the number of filters has been halved.

d) `double`: this experiment tests the effect on the model when *doubling* the number of available parameters with respect to `standard`; i.e. for each convolutional layer, the number of filters has been doubled.

e) `gates-off`: in this experiment, we test whether having gates has effects on the model's performance. To do this, for each gate element the values are set such that $\alpha^t_{i,c} = 0$ if $i \neq t$ and 1 otherwise; which is equivalent to having independent networks for each task. For a more fair comparison with `standard`, each task specific network is given more parameters; i.e. the base architecture is the one used in `double`.

The code uses Tensorflow 2.0 [1] and Scikit-learn [25]. All the models are trained from scratch every time.

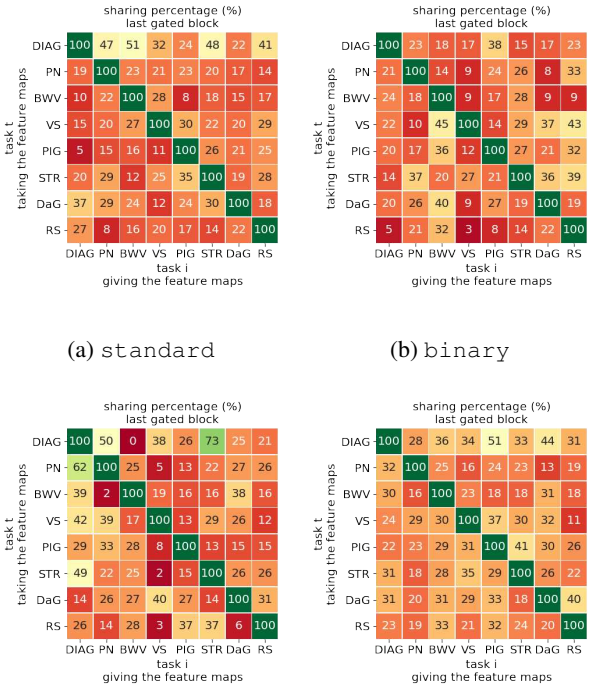
6. Results

We define *sharing fraction* between task t and task i at a certain gated block as

$$\text{SF}^t_i = \frac{1}{C}\sum_c^C \alpha^t_{i,c} \tag{11}$$

where C is the number of feature maps shared by task i. This number gives an indication of how many features coming from task i are being selected by task t.

It is known that the features learned at deeper layers of the network represent more abstract concepts related to the data, thus we are going to inspect what is being shared among the tasks at the last gated block. The sharing matrices for the four experiments are reported in Figure 4, where each element is evaluated through Eq. 11 times 100. Notably, in all the experiments the task that took more features from the others is the diagnosis task (DIAG). As a matter of fact, the diagnosis of the lesion should rely on the identification of the attributes, which would explain why this task requires more information from the others (i.e. the attributes). However, it appears that this does not apply when the labels in the diagnosis task are grouped into a binary task, as shown in Figure 4b. This might be because the *non*-melanoma images have very heterogeneous conformations, due to the varied nature of the lesions in this subset. Thus, the task might need to rely less on the information coming from other tasks. With the exception of `binary`, in the other experiments it seems that the correlation between diagnosis (DIAG) and attributes such as vascular structures (VS) and streaks (STR) is always present, with the latter reaching a high value of 73% in the `half` experiment (Figure 4c). These experiments also show a good correlation between DIAG and the attributes pigment network (PN), blue whitish veil (BWV) and vascular structures (VS), which are the three major criteria in the 7-point checklist method [4]. Surprisingly, `half` shows no reliance on the blue whitish veil (BWV) attribute for diagnosis at this stage, a fact that may be due to the higher correlation with STR. In this same experiment, there is a high correlation between PN and DIAG, which could hint at the fact that, given the limited number of parameters, the model has learned to base its de-



(a) `standard` (b) `binary`

(c) `half` (d) `double`

Figure 4: Sharing fraction matrices for the experiments detailed in Section 5, taken at the epoch with the best validation loss.

		DIAG	PN	BWV	VS	PIG	STR	DaG	RS	Avg. 7pt
standard	ac	45.8	55.4	85.1	63.5	62.5	49.1	48.6	65.1	61.3
	rc	45.5	53.2	76.5	49.9	53.8	52.4	50.6	67.2	57.7
	pr	40.3	53.9	75.7	44.0	53.7	46.0	49.8	63.5	55.2
half	ac	51.4	50.6	75.2	60.5	58.5	41.5	53.4	60.0	57.1
	rc	47.8	51.2	77.5	48.1	51.5	50.3	54.3	64.6	56.8
	pr	40.0	52.6	68.2	42.3	50.2	44.0	53.2	61.5	53.1
double	ac	45.8	53.7	71.6	45.1	57.0	54.4	53.2	66.1	57.3
	rc	41.0	53.5	79.4	42.2	52.5	52.6	53.7	66.4	57.2
	pr	37.5	53.4	68.4	38.1	51.5	48.3	52.8	63.1	53.7
gates-off	ac	44.3	44.8	77.2	47.8	43.3	42.8	47.3	56.7	51.4
	rc	38.5	47.7	79.8	47.5	46.5	53.9	49.0	65.0	55.6
	pr	35.3	49.8	70.0	42.0	42.9	46.4	48.3	62.4	51.7
binary	ac	77.2**	58.0	79.5	58.2	62.0	57.2	54.4	60.0	61.3
	rc	71.0 **	56.4	78.2	51.9	47.9	52.6	55.7	65.8	58.3
	pr	70.3 **	56.3	70.5	44.3	52.3	48.3	55.2	62.5	55.6
Kawahara et al. (*cbn*) *	ac	74.2	70.9	87.1	79.7	66.1	74.2	60.0	77.2	73.6
	rc	60.4	68.0	83.4	49.3	55.5	63.9	59.0	73.7	64.7
	pc	69.6	69.3	78.7	54.4	57.7	66.3	59.5	71.6	65.4

Table 2: Summary of the results for all the experiments. Last column indicates the average considering only the 7-point checklist criteria. (*) *cbn* refers to experiment *x-combine* [19], which uses additional data during training. (**) The DIAG task in `binary` only has 2 output classes, instead of 5. ac: accuracy; rc: recall; pc: precision.

cision on the pigment network through the diagnosis instead of the other way round (as would a human operator).

In all these experiments the values of the gates, i.e. $\alpha^t_{i,c}$, are free to move without regularization penalties and start with the same value. The fact that during training the values move towards both extremes of the spectrum, seems to indicate that indeed it is not necessary for the network to share every feature between the tasks and that this can give an indication of the decision mechanisms followed by the network. As a matter of fact, the model could have potentially learned to keep every gate to an intermediate value, as to have full access to all the feature maps for every classification tasks.

All the experiments clearly show the ability to fit the training data[2], with the training loss decreasing at a steady pace. However, it appears that the networks are prone to overfitting and poor generalization. This can be seen also in the quantitative results obtained on the unseen testing set, reported in Table 2. Performance in the various experiments is comparable, with minor differences. In particular, `standard` and `binary` are the best-performing setups overall, even though the latter deals with a simpler job due to the grouping in the DIAG task. Surprisingly, `half` performs slightly better than its counterparts on the DIAG task. An explanation for this might be found in the observation done in the paragraph above that DIAG has a SF score of 73% with STR. It might be that the DIAG task is somehow "stealing" feature maps from STR, whose performance does seem hindered. As a matter of fact, the tasks are all being trained together at the same time and they might be competing for resources when limited. On the other hand, the experiment `double` seems to have the opposite problem, and having more parameters its performance quickly shows signs of overfitting. Sharing features among tasks proved beneficial for the model, as clearly shown by the increase in performance between experiment `gates-off` and the ones in which gates are enabled. This gap is even more evident in the DIAG task, which is in fact known to be reliant on the other tasks. Furthermore, DIAG proves the hardest tasks to learn, probably because it represents a more abstract concept and because it is the only one with 5 possible labels, which are also highly unbalanced in the dataset. Compared to the performance of [19] (which uses the same data and split), our best-performing model (`standard`)

[2]Training curves available in supplementary material.

	GT		standard		binary			
inferred diagnosis	121	173	108	186	78	216	*0*	
	1	100	11	90	8	93	*1*	*true*
direct diagnosis	NA		281	13	246	48	*0*	
			76	25	42	59	*1*	
	0	*1*	*0*	*1*	*0*	*1*		
	predicted							

Table 3: Confusion matrices of the binary melanoma diagnosis for various experiments (0: not melanoma; 1: melanoma). Inferred diagnosis uses $\tau = 1$.

appears to have around a 10% gap in performance across the various metrics. However, for a fair comparison it has to be noted that the experiments in [19] make use of additional input features (such as metadata and clinical images) in their pipeline, providing additional information to the algorithm. Furthermore their base architecture is initialized on ImageNet and then re-trained, instead of being trained from scratch. Thus, a primary cause of these results might be the small size of the dataset, which makes it harder to train the algorithm from scratch and more prone to overfitting. In addition, looking at the training curves[2] also reveals how these tasks seem to have different speeds at which they start overfitting, suggesting that a different choice of training strategy might prove beneficial for this model.

The 7-point checklist method [4] allows to diagnose melanoma based on the detection of clinically significant attributes (Section 3). A score is assigned to each attribute based on its presence and/or irregularity. The scores are then summed together and a threshold τ determines whether the lesion is melanoma or not. In the following, we refer to the diagnosis obtained using the 7-point checklist rule over the attributes predicted by the network as *inferred diagnosis*. On the other hand, we refer to the output of the DIAG task as *direct diagnosis*. Table 3 summarizes the confusion matrices obtained through *inferred diagnosis* and *direct diagnosis* for the relevant experiments with a threshold $\tau = 1$[3]. It also reports the confusion matrix obtained by applying the 7-point checklist method on the ground truth (GT) labels. As this method only deals with a binary melanoma/non-melanoma classification, the 5 diagnosis labels have been grouped in 2 respective categories.

When applying the 7-point checklist method to the attribute predictions coming from the network, `standard` shows a performance that is similar that obtained through application of the same method on the GT labels, representing the clinical standard. Similarly, the `binary` experiment the model achieves a higher sensitivity on melanoma, but at the same time appears to provide many false positives. Looking at the ability of these models to detect melanoma based on the attributes or from direct diagnosis, enhances the differences between the `standard` and `binary` experiments. The former shows in fact more reliance on the attributes, which is proven by its high sensitivity to melanoma and ability to identify a relevant portion of the non-melanoma cases through this rule. Conversely, its direct diagnosis misses most of melanoma cases but rarely identifies a non-melanoma lesion as such. On the other hand, the model in `binary`, which is trained to directly identify melanoma, proves a better performance in its direct classification, while losing more precision when the 7-point checklist rule is applied to its attribute predictions.

[3]The confusion matrices for all the experiments and $\tau = 3$ can be found in the supplementary material.

7. Conclusion

The analysis of skin lesions inherently suits a MTL design due to the importance of detecting clinically significant attributes. Despite the good performance of "opaque" deep learning models, criteria driven by clinical knowledge are still preferred by the clinicians, e.g. the 7-point checklist addressed in this paper. Thus, this work has focused on developing a deep multi-task learning method that provides a simple way to improve interpretability of the learned associations between tasks. The proposed framework is based on the idea of learnable gates, which allow the model to learn which features to share among tasks. The values of these gates reveal how our model uses or mixes features from different tasks. Experiments have been carried out on the Derm7pt dataset, which provides annotations of the diagnosis and the 7-point checklist criteria for each lesion. The quantitative performance has proven behind the main related work [19], which only partly provides a fair comparison due to the use of additional data modalities in the pipeline. Nonetheless, the performance has shown satisfactory results when applying the 7-point checklist to the predicted attributes, approximating what is the clinical ground truth. Inspection of the associations learned by the model has revealed that in most experiments the diagnosis task is the one requiring more information from the other tasks (i.e. the checklist criteria). Notably, in our best-performing experiment there appears to be a higher sharing of the features of the major criteria, which also contribute more in the 7-point checklist rule. Overall, this work contributes to developing a tool that can be used to analyse the behaviour of a MTL CNN in a simple way and provide additional information to the end user, which could ultimately increase the potential of deep learning models for eventual clinical use.

Acknowledgements This work is partly supported by the Biomedical Research Council of the Agency for Science, Technology, and Research, Singapore. We thank Dr. K. Liang and Dr. M. Paknezhad of Bioinformatics Institute for the comments and insights in writing this manuscript.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016. 6

[2] Mundher Al-Shabi, Hwee Kuan Lee, and Maxine Tan. Gated-Dilated Networks for Lung Nodule Classification in CT Scans. *IEEE Access*, 7:178827–178838, 2019. 4

[3] Saeed Alzahrani, Waleed Al-Nuaimy, and Baidaa Al-Bander. Seven-Point Checklist with Convolutional Neural Networks for Melanoma Diagnosis. In *2019 8th European Workshop on Visual Information Processing (EUVIP)*, pages 211–216, Oct. 2019. 2

[4] Giuseppe Argenziano, Gabriella Fabbrocini, Paolo Carli, et al. Epiluminescence Microscopy for the Diagnosis of Doubtful Melanocytic Skin Lesions: Comparison of the ABCD Rule of Dermatoscopy and a New 7-Point Checklist Based on Pattern Analysis. *Arch Dermatol*, 134(12):1563–1570, Dec. 1998. 1, 3, 6, 8

[5] Giuseppe Argenziano, HP Soyer, V De Giorgi, Domenico Piccolo, Paolo Carli, and Mario Delfino. Interactive atlas of dermoscopy (book and CD-ROM). *EDRA Medical Publishing & New Media*, 2000. 2, 3

[6] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *arXiv:1910.10045 [cs]*, Dec. 2019. 1

[7] Alceu Bissoto, Fábio Perez, Vinícius Ribeiro, et al. Deep-Learning Ensembles for Skin-Lesion Segmentation, Analysis, Classification: RECOD Titans at ISIC Challenge 2018. *arXiv:1808.08480 [cs]*, Aug. 2018. 2

[8] Titus J. Brinker, Achim Hekler, Alexander H. Enk, et al. A convolutional neural network trained with dermoscopic images performed on par with 145 dermatologists in a clinical melanoma image classification task. *European Journal of Cancer*, 111:148–154, Apr. 2019. 1, 2

[9] P. Carli, E. Quercioli, S. Sestini, et al. Pattern analysis, not simplified algorithms, is the most reliable method for teaching dermoscopy for melanoma diagnosis to residents in dermatology. *British Journal of Dermatology*, 148(5):981–984, 2003. 1

[10] Rich Caruana. Multitask Learning. *Machine learning*, 28(1):41–75, 1997. 1, 2

[11] Eric Z. Chen, Xu Dong, Xiaoxiao Li, et al. Lesion Attributes Segmentation for Melanoma Detection with Multi-Task U-Net. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 485–488, Apr. 2019. 2

[12] Sheng Chen, Zhe Wang, Jianping Shi, et al. A multi-task framework with feature passing module for skin lesion classification and segmentation. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 1126–1129, Apr. 2018. 2

[13] Noel Codella, Veronica Rotemberg, Philipp Tschandl, et al. Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC). *arXiv:1902.03368 [cs]*, Mar. 2019. 2

[14] Vincent Dick, Christoph Sinz, Martina Mittlböck, et al. Accuracy of Computer-Aided Diagnosis of Melanoma: A Meta-analysis. *JAMA Dermatol*, 155(11):1291–1299, Nov. 2019. 1, 2

[15] Yuan Gao, Jiayi Ma, Mingbo Zhao, et al. NDDR-CNN: Layerwise Feature Fusing in Multi-Task CNNs by Neural Discriminative Dimensionality Reduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3205–3214, 2019. 2, 3, 4, 5

[16] Manu Goyal, Thomas Knackstedt, Shaofeng Yan, et al. Artificial Intelligence-Based Image Classification for Diagnosis of Skin Cancer: Challenges and Opportunities. *arXiv:1911.11872 [cs, eess, q-bio]*, Dec. 2019. 1, 2

[17] Achim Hekler, Jochen S. Utikal, Alexander H. Enk, et al. Superior skin cancer classification by the combination of human and artificial intelligence. *European Journal of Cancer*, 120:114–121, Oct. 2019. 2

[18] Sara Hosseinzadeh Kassani and Peyman Hosseinzadeh Kassani. A comparative study of deep learning architectures on melanoma detection. *Tissue and Cell*, 58:76–83, June 2019. 2

[19] Jeremy Kawahara, Sara Daneshvar, Giuseppe Argenziano, et al. Seven-Point Checklist and Skin Lesion Classification Using Multitask Multimodal Neural Nets. *IEEE Journal of Biomedical and Health Informatics*, 23(2):538–546, Mar. 2019. 2, 3, 5, 7, 8

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, et al. Focal Loss for Dense Object Detection. *arXiv:1708.02002 [cs]*, Aug. 2017. 5

[22] Aravindh Mahendran and Andrea Vedaldi. Understanding Deep Image Representations by Inverting Them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5188–5196, 2015. 1

[23] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, et al. Cross-Stitch Networks for Multi-task Learning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003, Las Vegas, NV, USA, June 2016. IEEE. 2

[24] Franz Nachbar, Wilhelm Stolz, Tanja Merkle, et al. The ABCD rule of dermatoscopy: High prospective value in the diagnosis of doubtful melanocytic skin lesions. *Journal of the American Academy of Dermatology*, 30(4):551–559, Apr. 1994. 1

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 6

[26] Sebastian Ruder. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv:1706.05098 [cs, stat]*, June 2017. 2

[27] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, et al. Latent Multi-Task Architecture Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4822–4829, July 2019. 2

[28] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, et al. Striving for Simplicity: The All Convolutional Net. *arXiv:1412.6806 [cs]*, Apr. 2015. 1

[29] Trevor Standley, Amir R. Zamir, Dawn Chen, et al. Which Tasks Should Be Learned Together in Multi-task Learning? *arXiv:1905.07553 [cs]*, May 2019. 2

[30] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Learning Task Relatedness in Multi-Task Learning for Images in Context. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval - ICMR '19*, pages 78–86, Ottawa ON, Canada, 2019. ACM Press. 2

[31] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Many Task Learning with Task Routing. *arXiv:1903.12117 [cs]*, Mar. 2019. 2, 3

[32] Saeid Asgari Taghanaki, Aicha Bentaieb, Anmol Sharma, S. Kevin Zhou, Yefeng Zheng, et al. Select, Attend, and Transfer: Light, Learnable Skip Connections. In *Machine Learning in Medical Imaging*, pages 417–425. Springer, Cham, Oct. 2019. 4

[33] P. Tschandl, G. Argenziano, M. Razmara, et al. Diagnostic accuracy of content-based dermatoscopic image retrieval with deep classification features. *British Journal of Dermatology*, 181(1):155–165, 2019. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/bjd.17189. 2

[34] Philipp Tschandl, Noel Codella, Bengü Nisa Akay, et al. Comparison of the accuracy of human readers versus machine-learning algorithms for pigmented skin lesion classification: An open, web-based, international, diagnostic study. *The Lancet Oncology*, 20(7):938–947, July 2019. 1

[35] Sulaiman Vesal, Shreyas Malakarjun Patil, Nishant Ravikumar, et al. A Multi-task Framework for Skin Lesion Detection and Segmentation. *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*, pages 285–293, Sept. 2018. 2

[36] Xulei Yang, Zeng Zeng, Si Yong Yeo, et al. A Novel Multi-task Deep Learning Model for Skin Lesion Segmentation and Classification. *arXiv:1703.01025 [cs]*, Mar. 2017. 2

[37] Yu Zhang and Qiang Yang. An overview of multi-task learning. *Natl Sci Rev*, 5(1):30–43, Jan. 2018. 2