NDDR-CNN: Layerwise Feature Fusing in Multi-Task CNNs by Neural Discriminative Dimensionality Reduction

Yuan Gao[1*] Jiayi Ma[2] Mingbo Zhao[3] Wei Liu[1*] Alan L. Yuille[4]
[1] Tencent AI Lab [2] Wuhan University [3] City University of Hong Kong [4] Johns Hopkins University
{ethan.y.gao, jyma2010, mbzhao4}@gmail.com, wl2223@columbia.edu, alan.yuille@jhu.edu

Abstract

*In this paper, we propose a novel Convolutional Neural Network (CNN) structure for general-purpose multi-task learning (MTL), which enables automatic feature fusing at every layer from different tasks. This is in contrast with the most widely used MTL CNN structures which empirically or heuristically share features on some specific layers (*e.g.*, share all the features except the last convolutional layer). The proposed layerwise feature fusing scheme is formulated by combining existing CNN components in a novel way, with clear mathematical interpretability as discriminative dimensionality reduction, which is referred to as Neural Discriminative Dimensionality Reduction (NDDR). Specifically, we first concatenate features with the same spatial resolution from different tasks according to their channel dimension. Then, we show that the discriminative dimensionality reduction can be fulfilled by 1×1 Convolution, Batch Normalization, and Weight Decay in one CNN. The use of existing CNN components ensures the end-to-end training and the extensibility of the proposed NDDR layer to various state-of-the-art CNN architectures in a "plug-and-play" manner. The detailed ablation analysis shows that the proposed NDDR layer is easy to train and also robust to different hyperparameters. Experiments on different task sets with various base network architectures demonstrate the promising performance and desirable generalizability of our proposed method. The code of our paper is available at* https://github.com/ethanygao/NDDR-CNN.

1. Introduction

Deep convolutional neural networks (CNNs) have greatly pushed the previous limits of various computer vision tasks since the seminal work [20] in 2012. CNN models can naturally integrate hierarchical features and classifiers, which can be trained in an end-to-end manner. Benefiting from that, significant improvements have been witnessed in fundamental computer vision tasks, such as image classification [14–16, 20, 45], object detection [8, 9, 13, 25, 35, 36, 42–44, 47, 48, 50], semantic segmentation [1–5, 27, 33, 37, 51], *etc*.

One of the main factors that can further boost the CNN performance is multi-task learning (MTL), which is engaged in learning multiple related tasks simultaneously. This is because related tasks can benefit from each other by jointly learning certain shared, or more precisely, mutually related representations [13, 19]. The multiple supervision signals originating from different tasks in MTL can be viewed as implicit data augmentation (on labels) or additional regularization (among different tasks) [39]. This enables to learn mutually related representations that work well for multiple tasks, thus avoiding overfitting and leading to better generalizability.

Most commonly, the CNN structure for MTL is heuristically determined by sharing all convolutional layers, and splitting at fully-connected layers for task-specific losses. However, as different layers learn low-, mid-, and high-level features [57], a natural question arises: *Why would we assume that the low- and mid-level features for different tasks in MTL should be identical, especially when the tasks are loosely related? If not, is it optimal to share the features until the last convolutional layer?*

The study in Misra *et al*. [31] reveals that sharing/splitting at different layers gives different performances. Especially, improper features sharing at some layers may degrade the performance of some, or even all, tasks. In addition, the deep nature of CNNs makes it infeasible to exhaustively test all the possible structures to find the optimal sharing/splitting scheme. In order to tackle this issue, Misra *et al*. used trainable *scalars* to weighted-sum the features from different tasks at multiple CNN levels and achieved state-of-the-art performance [31].

We consider this problem in another way, *i.e*., by leveraging all the hierarchical features from different tasks. This is because that the CNN layers trained by different tasks can be treated as different feature descriptors, therefore the fea-

* indicates corresponding authors.

tures learned from them can be treated as different representations/views of input data. *We hypothesize that these features, obtained from multiple feature descriptors (*i.e.*, different CNN levels from multiple tasks), contain additional discriminative information of input data, which should be exploited in MTL towards better performance*.

Specifically, starting with K single-task networks (from K tasks), a direct attempt to take advantage of hierarchical features from all the tasks is that: we may concatenate all the task-specific features with the same spatial resolution *from different tasks* according to the feature *channel* dimension. After that, we expect the CNN to learn a discriminative feature embedding for each task, by receiving these concatenated features as inputs. However, most existing CNNs have carefully designed structures, which only receive features (tensors) with a fixed number of feature channels. By concatenating features, we substantially enlarge the number of channels as K times if we have K tasks. This makes it impossible to feed these concatenated features to the following layers of the CNN.

This property of the CNN motivates us to conduct *discriminative dimensionality reduction* on the concatenated features. Its purpose is to learn a discriminative feature embedding, and to reduce the feature dimension such that it can satisfy the input channel requirement of the following layers. Feature transformation is one of the most important approaches to tackle the discriminative dimension reduction problem. It aims to learn a projection matrix that projects the original high-dimensional features into a low-dimensional representation, while keeping as much discriminative information as possible.

In this paper, we show that, from the perspective of feature transformation, discriminative dimensionality reduction is closely related to some common operations of modern CNNs. Specifically, the transformation in discriminative dimensionality reduction is in fact equivalent to the 1×1 *convolution*. In addition, the constraints on the norm of the transformation weights (*i.e.*, the weights of the 1×1 convolutional layer) and input feature vectors can be represented by *weight decay* and *batch normalization* [17], respectively. We refer to the combination of these operations as *Neural Discriminative Dimensionality Reduction* (NDDR). Therefore, we are able to link the original single-task networks from different tasks by the NDDR layers. Desirably, the proposed network structure can be trained end-to-end in the CNN without any extraordinary operations.

It is worth noting that this paper focuses on a general structure for general-purpose MTL. The proposed NDDR layer combines existing CNN components in a novel way, which possesses clear mathematical interpretability as discriminative dimensionality reduction. Moreover, the use of the existing CNN components is desirable to guarantee the extensibility of our method to various state-of-the-art CNN architectures, where the proposed NDDR layer can be used in a "plug-and-play" manner. The rest of this paper is organized as follows. First, we describe the NDDR layer and propose a novel NDDR-CNN as well as its variant NDDR-CNN-Shortcut for MTL in Sect. 3. After that, we discuss the related works in Sect. 2, where we show that our method can generalize several state-of-the-art methods, which can be treated as our special cases. In Sect. 4, the ablation analysis is performed, where the hyperparameters used in our network are suggested. Following that, the experiments are performed on different network structures and different task sets in Sect. 5, demonstrating the promising performance and desirable generalizability of our proposed method. We make concluding remarks in Sect. 6.

2. Related Works

Various computer vision tasks benefit from MTL [41], such as detection [8,9,13,36,42–44,50], human pose and semantic segmentation [52], surface normal prediction, depth prediction, semantic segmentation [6], action recognition [53, 54], *etc*. Several human face related tasks, including face landmark detection, attributes detection (such as smile and glasses), gender classification, and face orientation, were studied in [12,34,49]. Yim *et al*. used face alignment and reconstruction as auxiliary tasks for face recognition [56]. MTL on sequential data was also studied in [24]. Recently, Kokkinos proposed a UberNet which enables a great number of low-, mid-, and high-level vision tasks to be handled simultaneously [19].

CNN based MTL theory has also been greatly developed in recent years. Long and Wang proposed a deep relationship network to enable the feature sharing at the fully-connected layers [28]. Starting with a thin network, a top-down layerwise widening method was proposed to automatically determine which layer to split [29]. Yang and Hospedales used tensor decomposition at initialization to share the MTL weights [55]. The weights to combine the task-specific losses were also studied, and a Bayesian approach was proposed to predict these weights [18]. The cross-stitch network used *trainable scalars* to fuse (*i.e.*, weighted sum) the features at layers in the same level from different tasks [31]. Most recently, the sluice network predefines several subspaces on the features from each task and learns the weights to fuse the features across different subspaces [40].

Our method is also related to discriminative dimensionality reduction. The goal of the discriminative dimensionality reduction techniques is to reduce the computational and storage costs, by learning a low-dimensional embedding that retains most of the discriminative information. Linear discriminant analysis (LDA) is one of the most popular conventional discriminative dimensionality reduction methods, which aims to seek the optimal projection matrix by maxi-

mizing the between-class variance and meanwhile minimizing the within-class variance [30]. In addition, low-rank metric learning [26] can also be viewed as a discriminative dimensionality reduction technique.

Introduced by network in network [22], 1×1 convolution has been widely used in many modern CNN architectures [14, 16, 23, 46]. For example, it was used in ResNet to reduce the number of weights to train, by producing a "bottleneck unit" [14]. 1×1 convolution is also implemented in the feature pyramid network to fuse hierarchical features (in different CNN levels) on a single task [23]. Note that we do NOT claim the 1×1 convolution as our novelty. Instead, we use 1×1 convolution together with batch normalization and weight decay in a novel way, which yields an NDDR layer. In other words, we formulate the multi-task feature fusing paradigm as a discriminative dimensionality reduction problem, and use the NDDR layer, which is composed of 1×1 convolution, batch normalization, and weight decay, to learn the feature embeddings from multiple tasks. The use of the existing CNN components ensures the extensibility of our method to various state-of-the-art CNN architectures in a "plug-and-play" manner.

3. Methodology

In this section, we propose a novel method to automatically learn the optimal structure for layerwise feature fusing in a multi-task CNN. Instead of the "split-style" multi-task CNN (*e.g.*, split at the last convolutional layer for different task-specific losses), we consider the "fuse-style" network combining multiple single-task networks via discriminative dimensionality reduction.

We first relate the discriminative dimensionality reduction to 1×1 convolution and propose the NDDR layer. Then, a novel multi-task network is proposed, namely NDDR-CNN, where the NDDR layer is leveraged to connect the original single-task networks. Moreover, a variant of NDDR-CNN is introduced, namely NDDR-CNN-Shortcut, which enables to directly route the gradients to the lower NDDR layers by shortcut connections. Finally, we give the implementation details of the proposed network.

3.1. NDDR Layer

As discussed in previous sections, we aim to utilize the hierarchical features learned from different tasks. It is unlike the most widely used method which heuristically shares all the low-(and mid-) level features and splits the network at the last convolutional layer.

In order to do that, we first concatenate the task-specific features from different tasks according to the channel dimension. Then, we use a discriminative dimensionality reduction technique to reduce the feature channels such that the output features satisfy the channel dimension requirement of the next CNN layers. We refer to the new CNN layer with such operations as the Neural Discriminative Dimensionality Reduction (NDDR) layer.

Specifically, let $F_l^i \in \mathbb{R}^{N \times H \times W \times C}$ be the output features (arranged in a tensor) at an intermediate layer l of task i. Regarding K tasks, concatenating the features from them according to the channel dimension gives:

$$F_l = [F_l^1, ..., F_l^K] \in \mathbb{R}^{N \times H \times W \times KC}. \tag{1}$$

Discriminative dimensionality reduction learns a transformation W to reduce the dimensionality of the input features, while keeping most discriminative information:

$$F_l^{i*} = F_l W^i, \tag{2}$$

where $W^i \in \mathbb{R}^{KC \times M}$ and $M < KC$ is the projection matrix to be learned for each task i. In our case, M is equal to C (*i.e.*, $F_l^{i*} \in \mathbb{R}^{N \times H \times W \times C}$) in order to satisfy the channel size requirement of the following CNN layers.

Conventional discriminative dimensionality reduction methods learn the transformation W with specific assumptions/objectives which make the features more separable. For example, Linear Discriminative Analysis (LDA) learns W by minimizing the projected *within-class* variation and meanwhile maximizing the projected *between-class* variation [30]. Intuitively, the objective function of the discriminative dimensionality reduction is related to the CNN loss, *i.e.*, the features projected by discriminative dimensionality reduction are more separable, therefore giving a smaller CNN loss.

Motivated by this, we aim to learn the transformation W in the CNN implicitly by back-propagation. The transformation $W \in \mathbb{R}^{KC \times C}$ can be represented precisely by a convolution operation with stride 1 and size ($C \times 1 \times 1 \times KC$), where these size dimensions represent filters, kernel height, kernel width, and channels, respectively. It is worth noting that the convolution with 1×1 kernel size and 1 stride enables to perform the computations only according to channels, rather than fusing the features at different spatial locations or changing the spatial sizes of the features.

In addition, discriminative dimensionality reduction methods also have constraints on the norms of the transformation W (to avoid a trivial solution) and the input features F_l (otherwise, the learned projections may project the features to some *noise* directions). We borrow this idea to our NDDR layer for stable learning, which can be achieved by imposing *batch normalization* on the input features and ℓ_2 *weight decay* on the 1×1 convolutional weights W, respectively.

In summary, a novel NDDR layer is proposed in this section. The NDDR layer can be constructed by: 1) concatenating the task-specific features with the same spatial resolution from different tasks according to the channel dimension, and 2) using 1×1 *convolution* to learn a discriminative

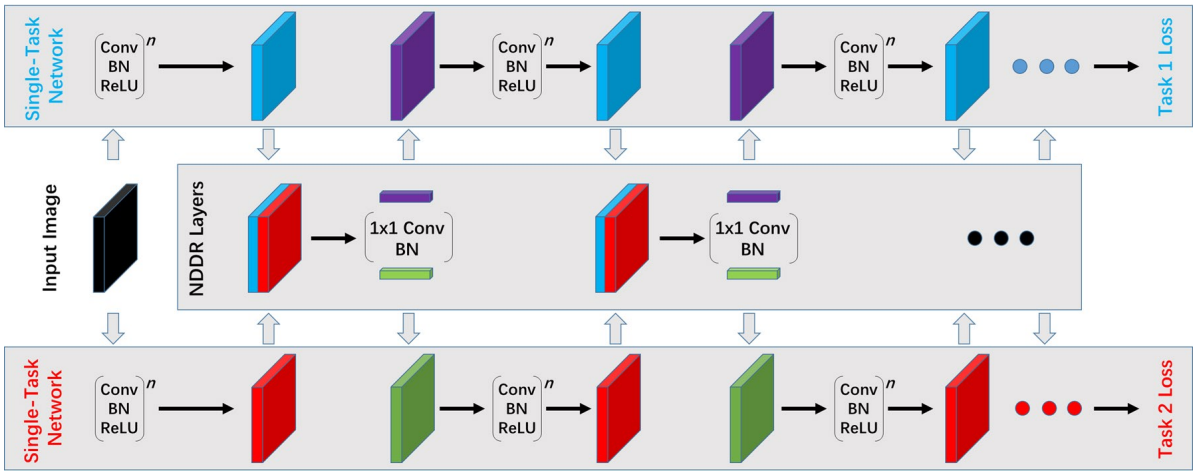

Figure 1. The network structure of NDDR-CNN. In the NDDR layer, we concatenate the outputs of original single-task networks from multiple tasks (two tasks shown here), and use 1×1 convolution to perform discriminative dimensionality reduction. Therefore, the output of the NDDR layer retains the discriminative information from both the input features, and can be fed to the following layers of the single-task networks. The proposed NDDR layer can be leveraged to connect the original single-task networks of multiple levels for layerwise feature fusing (best view in color).

feature embedding for each task. We also use *batch normalization* on the input features of the NDDR layer for stable learning. We train the NDDR layer by back-propagating the *task-specific* losses and the ℓ_2 *weight decay* loss on the 1×1 convolutional weights W. *Without any extraordinary operations, the network with our NDDR layer can be trained in an end-to-end fashion.*

3.2. NDDR-CNN Network

We insert the NDDR layers in multiple levels of the original single-task networks, to enable layerwise feature fusing/embedding for different tasks. We refer to the proposed network for MTL as the NDDR-CNN network.

Figure 1 shows the NDDR-CNN network structure for two tasks. It can easily be extended to K-task problems. Let the number of channels for the single-task features be D. Then NDDR-CNN for K tasks can be constructed by: 1) concatenating the features from K tasks according to the *channel* dimension, and 2) using 1×1 convolution with (filters $\times 1 \times 1 \times$ channels) = $(C \times 1 \times 1 \times KC)$ to conduct dimensionality reduction, where C is the channel dimension size of the output features from each task.

Note that the elements of the NDDR layer are common CNN operations, which ensures that the proposed NDDR layer can be extended to various state-of-the-art CNN architectures in a "plug-and-play" manner.

3.3. NDDR-CNN Network with Shortcuts

In order to avoid gradient vanishing at lower NDDR layers, we propose a new network that enables to pass gradients directly from the last convolutional layer to the lower ones via *shortcut connections*, namely NDDR-CNN-Shortcut.

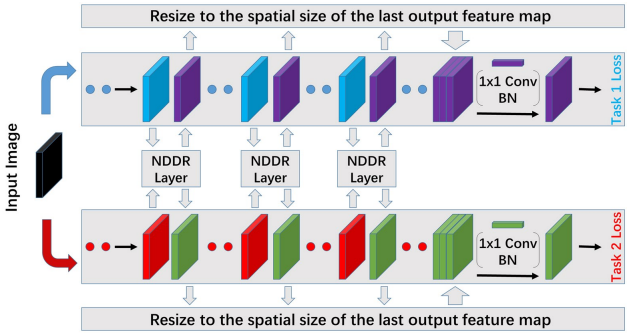

Figure 2. The NDDR-CNN-Shortcut network. In NDDR-CNN-Shortcut, we use shortcut connections to enable gradients to directly route to the lower NDDR layers. This is done by resizing the lower NDDR output to the spatial size of the last NDDR output, then concatenating the resized features of the same task according to the channel dimension, and finally using 1×1 convolution to do dimensionality reduction (best view in color).

Specifically, the output of each NDDR layer is resized to the spatial sizes of the last convolutional output. Then we concatenate all the resized feature maps of the same task from different layers together according to the channel dimension. Finally, in order to fit the input size of the following fully-convolutional/connected layers, we further use 1×1 convolution to learn more compact feature tensors (*e.g.*, in the VGG network, we reduce the channel dimension of concatenated features to 512). An illustration of the NDDR-CNN-Shortcut network is shown in Fig. 2.

3.4. Relationship to State-of-the-art Methods

Our method is closely related to the cross-stitch network [31]. In order to seek the optimal network struc-

ture for MTL, the cross-stitch network [31] uses *trainable scalars* to scale the features at layers in the same level from different tasks, and then adds them together as new features. Our work is related to the cross-stitch network but has three major differences: 1) We have different motivations, *i.e*., our work is motivated by learning discriminative low-dimensional embeddings on the concatenated features from multiple tasks. 2) Our method can generalize the cross-stitch network *by fixing the off-diagonal elements of the projection matrix to 0, and only updating the diagonal elements with the same value* (*i.e*., update the projection matrix by α and β in Eq. (3)). 3) We further propose an NDDR-CNN-Shortcut model, which further uses hierarchical features from different CNN levels for better training and convergence. Similarly, our network also takes the sluice network [40] as a special case: the sluice network predefines a fixed number of subspaces to fuse the features from different tasks between different subspaces (each contains multiple feature channels), while our model can automatically fuse the features according to *each single channel*.

3.5. Implementation Details

Note that the state-of-the-art convolutional network architectures such as VGG [45], ResNet [14], and DenseNet [16] typically group similar operations into stages/blocks, where each stage contains $\{\text{convolution-activation}\}^n$ (possibly with pooling). In order to make the least modification to the baseline network architecture to investigate the performance of the proposed NDDR layer, we only connect the two networks by applying the NDDR layer *at the end of each stage/block*. For example, we apply the NDDR layers at the outputs of *pool1, pool2, pool3, pool4,* and *pool5* for the VGG network. Similarly, as much deeper as ResNet is, we still apply only 5 NDDR layers in it, *e.g*., at the outputs of *conv1n3, conv2_3n3, conv3_4n3, conv4_6n3,* and *conv5_3n3* for ResNet-101. Also, it is worth noting that the additional parameters introduced by the NDDR layers are also very few with respect to those for the whole networks. For example, when applying NDDR layers at *pool1, pool2, pool3, pool4,* and *pool5* of the VGG-16 network, the additional parameters for the NDDR layers are only 1.2M, being 0.8% compared to the original 138M parameters of the entire VGG-16.

4. Ablation Analysis

In this section, several ablations have been done to analyze NDDR-CNN. Two factors about the NDDR layers are analyzed, *i.e*., different 1×1 convolutional weight initializations, and the scales of the "base" learning rate (*i.e*., the learning rate for the remaining network) to train the NDDR layers. We also analyze which pretrained weights should be used as initialization, *i.e*., the weights trained on ImageNet or different single tasks. We use a two-task problem here and in the following sections. For the ablation analysis, we use *semantic segmentation* and *surface normal prediction*.

Dataset. The NYU v2 dataset [32] is used for semantic segmentation and surface normal prediction. We use the official train/val splits which include 795 images for training and 654 images for validation. For semantic segmentation, the NYU v2 dataset contains 40 classes such as beds, cabinets, clothes, books, *etc*. [11]. The NYU v2 dataset also has the pixel-level surface normal ground-truths precomputed by the depth labeling [6, 21, 32].

Network Architecture. We use the state-of-the-art architecture for pixel-level tasks, *i.e*., Deeplab [4], for both semantic segmentation and surface normal prediction. Deeplab is essentially a VGG or ResNet network backbone with atrous convolution and atrous spatial pyramid pooling. We do not implement Fully Connected CRFs or multi-scale inputs as they are not related to the NDDR layer we proposed. We are careful to stick closely to the proposed NDDR layer by using the same atrous convolution and atrous spatial pyramid pooling for all the methods, so as to clearly see the effects of simply incorporating the NDDR layer. We use the Deeplab-VGG-16 architecture in all the ablation analysis.

Losses. We use the softmax cross-entropy loss for semantic segmentation. For surface normal prediction, we use the ℓ_2 regression loss after normalizing the normal vector of each pixel to have unit ℓ_2 norm (*i.e*., this represents a direction for a certain angle). Therefore, our loss for surface normal prediction is also equivalent to the cosine loss.

Evaluation Metrics. The performance of semantic segmentation is evaluated by mean Intersection over Union (mIoU) and Pixel Accuracy (PAcc). For surface normal estimation, we use Mean and Median angle distances of all the pixels for evaluation (the lower the better). In addition, we also use the metrics introduced by [6], which are the percentage of pixels that are within the angles of $11^\circ, 22.5^\circ, 30^\circ$ to the ground-truth (the higher the better).

4.1. Initializations for NDDR Layers

In order to have a mild initialization which resembles single-task networks, we keep the diagonal weights of the NDDR layer as non-zeros. Recall that the NDDR layer for a two-task problem is $F^{out} = \left[F_1^{in}, F_2^{in}\right]\left[W_1^\top, \ W_2^\top\right]^\top$. In order to initialize the NDDR weights W_1 and W_2, we let:

$$F_1^{out} = \left[F_1^{in}, F_2^{in}\right] \begin{bmatrix} \alpha & 0 & \dots & 0 & \beta & 0 & \dots & 0 \\ 0 & \alpha & \dots & 0 & 0 & \beta & \dots & 0 \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & \alpha & 0 & 0 & \dots & \beta \end{bmatrix}^\top, \quad (3)$$

where F_1^{in}, F_2^{in} are the inputs to the NDDR layer and F_1^{out} is the output which will be fed to Task 1[1]. By writing the weight of NDDR in this way, it shows that if we initialize

[1]We take an NDDR layer from one task as an example, and the initialization of the NDDR layer for the other task is identical.

	Surface Normal Prediction					Semantic Seg.	
	Angle Distance (Lower Better)		Within t° (%) (Higher Better)			(%) (Higher Better)	
(α, β)	Mean	Med.	11.25	22.5	30	mIoU	PAcc
$(1, 0)$	14.0	10.3	53.2	79.1	88.6	**36.2**	66.5
$(0.9, 0.1)$	**13.9**	**10.2**	**53.5**	**79.5**	**88.8**	**36.2**	**66.4**
$(0.5, 0.5)$	**13.9**	**10.2**	**53.5**	79.3	88.6	36.0	**66.4**
$(0.1, 0.9)$	14.3	10.6	52.4	78.5	88.0	35.7	66.1
$(0, 1)$	14.2	10.6	52.5	78.2	87.8	35.7	65.9
Random	15.0	11.6	49.0	76.7	87.0	33.4	64.4

Table 1. The results with different initializations for the **NDDR layers**.

	Surface Normal Prediction					Semantic Seg.	
	Errors (Lower Better)		Within t° (%) (Higher Better)			(%) (Higher Better)	
Scale	Mean	Med.	11.25	22.5	30	mIoU	PAcc
1	14.7	11.2	50.1	77.3	87.4	35.9	65.9
10	14.4	10.7	51.9	78.1	87.9	36.0	66.1
10^2	**13.9**	**10.2**	**53.5**	79.5	88.8	**36.2**	**66.4**
10^3	**13.9**	10.6	52.4	**79.6**	**89.2**	35.7	**66.4**

Table 2. The results with different learning rates for the **NDDR layers** (*i.e.*, the scale with respect to the base learning rate for other layers). The learning rates are represented as different **scales** with respect to those for other perception convolutional layers.

$\alpha = 1$ and $\beta = 0$, the whole network will start with the single-task networks, *i.e.*, $F_1^{out} = F_1^{in}$. We refer to this as *diagonal initialization*.

In the experiments, we have 5 different diagonal initializations with (α, β) ranging from $(1, 0)$ to $(0, 1)$, *i.e.*, from the mildest initialization from the same tasks to the most severe initialization from the opposite tasks. In addition, we also discuss the random initialization of the whole weight matrices $[W_1^\top, W_2^\top]$ with Xavier initialization [10].

Table 1 shows the performance with different initializations of the NDDR layer. The results show that the *diagonal initialization* is better than Xavier initialization[2], and that the initialization of (α, β) has a little effect on results. In the following experiments, we use diagonal initialization with $(\alpha, \beta) = (0.9, 0.1)$.

4.2. Learning Rates for NDDR Layer

In this section, we discuss the learning rate for the NDDR layer. There are two main reasons to set a larger learning rate specifically for the NDDR layer. First, as analyzed in Sect. 4.1, the NDDR-CNN becomes single-task networks if we set a very large weight (*e.g.*, $\alpha = 1$, much larger than the weights of perception convolutional layers) at the diagonal of $W_1^\top$. Thus, we hypothesize that the magnitude of the NDDR layer weights should be larger, therefore requiring a larger learning rate. Second, a larger learning rate for NDDR layers is also necessary if we fine-tune the NDDR-CNN from the pretrained single-task networks.

[2]Note that the results from Xavier initialization (in Table 1) are still comparable with the previous state-of-the-art method (*i.e.* the cross-stitch network and the sluice network in Table 4) in surface normal prediction.

	Surface Normal Prediction					Semantic Seg.	
	Errors (Lower Better)		Within t° (%) (Higher Better)			(%) (Higher Better)	
Init.	Mean	Med.	11.25	22.5	30	mIoU	PAcc
Pret.	14.3	10.6	52.2	78.6	88.2	34.3	65.2
Sing.	**13.9**	**10.2**	**53.5**	**79.5**	**88.8**	**36.2**	**66.4**

Table 3. The results with different pretrained models. **Pret.** means the pretrained Deeplab-VGG-16 weights for semantic segmentation on Pascal VOC 2012, and **Sing.** represents the finetuned weights from the corresponding target single tasks (through single-task networks).

Therefore, we analyze the proper learning rate for the NDDR layer as how many times it should be with respect to the base learning rate (for the remaining network excluding the NDDR layers). Table 2 shows the performance of using different learning rates for the NDDR layer. It verifies that larger learning rates should be applied for NDDR layers. In the following experiments, we use 100 times of the base learning rate for the NDDR layer.

4.3. Pretrained Weights for Network Initialization

Two network initialization strategies can be applied. We may use the network weights pretrained on a general task (*e.g.* pretrained Deeplab-VGG-16 [4] for semantic segmentation on Pascal VOC 2012 [7]) or finetuned on corresponding target single tasks. The results with different pretrained models are summarized in Table 3, which show that initializing the finetuned weights from target single tasks performs better. The results indicate that by simply adding several NDDR layers, we have enlarged the capability of the (converged) original networks, which further enables to skip the previously existing saddle points.

5. Experiments

In this section, we perform various experiments on both different network structures and different task sets to demonstrate the promising performance and desirable generalizability of the proposed NDDR-CNN.

Specifically, VGG-16 [45] and ResNet-101 [14] have been used in our experiments, we put the results on AlexNet [20] in Table S1 of the Supplementary Materials. In addition, we also test our proposed NDDR-CNN-Shortcut with the VGG structure, where the gradients can be passed to the lower NDDR layers by the shortcut connections. This can further demonstrate the performance of the proposed NDDR layer. We refer to this network as *VGG-16-Shortcut*.

For evaluation, we train each task separately using the common single-task network architecture without NDDR layers as our **single task baseline**. The results from the most widely used heuristic multi-task network structure are performed as our **multi-task baseline**, where all the convolutional layers are shared and the split takes place after

	Surface Normal Prediction					Semantic Seg.	
	Errors (Lower Better)		Within t° (%) (Higher Better)			(%) (Higher Better)	
	Mean	Med.	11.25	22.5	30	mIoU	PAcc
Sing.	15.6	12.3	46.4	75.5	86.5	33.5	64.1
Mul.	15.2	11.7	48.4	76.2	87.0	33.4	64.2
C.-S.	15.2	11.7	48.6	76.0	86.5	34.8	65.0
Sluice	14.8	11.3	49.7	77.1	88.0	34.9	65.2
Ours	**13.9**	**10.2**	**53.5**	**79.5**	**88.8**	**36.2**	**66.4**

Table 4. Experimental results on semantic segmentation and surface normal prediction using **VGG-16**. Sing., Mul., C.-S., and Sluice represent the single-task baseline, the multiple-task baseline, the cross-stitch network, and the sluice network, respectively.

	Surface Normal Prediction					Semantic Seg.	
	Errors (Lower Better)		Within t° (%) (Higher Better)			(%) (Higher Better)	
	Mean	Med.	11.25	22.5	30	mIoU	PAcc
Sing.	15.6	12.7	44.3	74.8	87.2	39.5	69.2
Mult.	16.3	13.8	41.1	73.9	86.5	39.1	68.7
C.-S.	15.9	13.2	42.9	75.1	86.8	40.5	69.5
Sluice	15.3	12.8	44.1	76.9	88.2	40.8	70.1
Ours	**14.4**	**11.6**	**48.5**	**79.1**	**89.5**	**43.3**	**71.5**

Table 5. Experimental results on semantic segmentation and surface normal prediction using **ResNet-101**. Sing., Mul., C.-S., and Sluice represent the single-task baseline, the multiple-task baseline, the cross-stitch network, and the sluice network, respectively.

	Surface Normal Prediction					Semantic Seg.	
	Errors (Lower Better)		Within t° (%) (Higher Better)			(%) (Higher Better)	
	Mean	Med.	11.25	22.5	30	mIoU	PAcc
Sing.	15.5	12.3	46.3	75.5	86.5	33.5	64.4
Mult.	15.2	11.8	48.3	76.1	86.6	33.6	64.4
C.-S.	14.8	11.1	50.3	76.9	87.0	35.0	65.1
Sluice	14.2	10.6	51.7	78.2	88.2	35.3	65.3
Ours	**13.5**	**9.8**	**55.3**	**80.5**	**89.3**	**36.7**	**67.0**

Table 6. Experimental results on semantic segmentation and surface normal prediction using **VGG-16-Shortcut**. Sing., Mul., C.-S., and Sluice represent the single-task baseline, the multiple-task baseline, the cross-stitch network, and the sluice network, respectively.

the last convolutional layer. We also investigate the performances of the **cross-stitch network** [31] and the state-of-the-art **sluice network** [40] for comparison, in which we apply the same number of cross-stitch/sluice layers at the same locations as our NDDR layers. We use the number of subspaces as 2 for sluice network as suggested in [40]. For the fair comparison, we use the best hyperparameters in [31] and [40] to train the corresponding networks[3].

As we aim to a general purpose MTL method, very diverse task sets are chosen to evaluate our performance. These include *pixel-level labeling tasks on scene images*, *i.e.*, semantic segmentation and surface normal prediction, and *image-level classification tasks on human faces*, *i.e.*, age and gender classification. In the following subsections, we perform the semantic segmentation and surface normal prediction on NYU v2 dataset [32], and the age and gender classification on the IMDB-WIKI dataset [38]. We detail the task configurations in the following.

5.1. Semantic Segmentation and Surface Normal Prediction

In this section, we test our network on VGG-16, ResNet-101, and VGG-16-Shortcut to verify the desirable performance of the proposed network. In addition, by doing this, we further demonstrate the desirable generalizability of the proposed NDDR layers on different network architectures.

The configurations of the semantic segmentation, surface normal prediction, and the best hyperparameters to train the network can be found in Sect. 4.

5.1.1 Experiments on VGG-16 Network

In this section, we combine two VGG-16 networks by applying the NDDR layer at the outputs of *pool1, pool2, pool3, pool4* and *pool5*.

Table 4 shows the results on semantic segmentation and surface normal prediction using the VGG-16 network. Though as simple as our method is, it significantly outperforms the state-of-the-art methods. For example, our method outperforms the sluice network by around 3.8% in "within 11.25°" metric in surface normal prediction, and 1.1%-1.2% for both metrics in semantic segmentation. These results demonstrate the promising performance of our method.

5.1.2 Experiments on ResNet-101 Network

We perform the NDDR layers in the ResNet-101 network, where the NDDR layers are only applied at the output of *conv1n3, conv2_3n3, conv3_4n3, conv4_6n3* and *conv5_3n3*.

The results are shown in Table 5. It indicates that our method consistently outperforms the baseline and state-of-the-art results. Noted that comparing with the as deep as 101-layer network, we only slightly modified the ResNet-101 by adding *five* NDDR layers. These results further demonstrate the efficacy of the proposed NDDR layer.

5.1.3 Experiments on VGG-16 Network with Shortcut Connections

We test the proposed NDDR-CNN-Shortcut with the VGG-16 structure, *i.e.*, the VGG-16-Shortcut network to further validate our performance.

Compared with ResNet, the VGG-16-Shortcut network resembles more to DenseNet [16]. In VGG-16-Shortcut,

[3]We show that the hyperparameters, originally from AlexNet in [31], are still the best for other network backbones. Please see Table S2 in the Supplementary Materials.

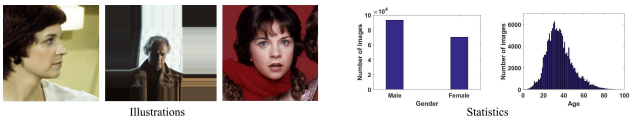

Figure 3. Some example illustrations and statistics of ages and genders for the IMDB-WIKI dataset. The statistics show that we have sufficient samples to train both genders, and the ages of most samples are between 20 - 50.

the gradients can be passed to the lower NDDR layers by the *direct and shortest* shortcut connections, rather than by *multiple* shortcuts in ResNet-like networks where the gradients may still decay[4].

The results for VGG-16-Shortcut are shown in Table 6. Compared with the performance on the "vanilla" VGG-16 network (*i.e.*, Table 4), the results of all the methods are improved in VGG-16-Shortcut. Especially, the improvements in our method are higher than those in our counterpart.

Table 6 shows that our method consistently outperforms the state-of-the-art methods. Especially, our method outperforms the sluice network by 3.1% for "within 11.25°" metric in surface normal prediction, and 1.0%-1.5% for the two metrics in semantic segmentation.

5.2. Age Estimation and Gender Classification

Dataset. We use the IMDB-WIKI dataset [38] for this task set, which contains 460723 images collected from 20284 subjects. After filtering out the images with more than one faces and the images without age or gender labels, the remaining 187103 images from 12325 subjects are used to perform our experiments. These contain images for both genders with ages from 0 to 99. We randomly choose 24090 images from 2000 subjects for evaluation, and the remaining 163013 images from 10325 subjects are used for training. In the training set, we have sufficient samples for both male and female, but the training data for ages is imbalanced. Some image examples, with the gender and age statistics, are shown in Fig. 3.

Network Architecture. The VGG-16 network is used as the base network in this experiment, with the NDDR layers applied after *pool1, pool2, pool3, pool4* and *pool5*.

Losses. Motivated by [38], we treat both age and gender estimations as classification problems, *i.e.*, 2-class and 100-class classifications. We use softmax cross-entropy loss in both tasks.

Evaluation Metrics. Classification accuracy (Acc) is used to evaluate the gender classification. For age estimation, we follow the metric from [38]. That is, for each image i, we treat the output $p_i \in \mathbb{R}^{100}$ from softmax as the probabilities for different ages (*i.e.*, 0-99). Therefore the final age estimation is calculated by $\text{age}_i^* = \sum_{k=0}^{99} p_i(k)\text{dict}(k)$, where dict = $\{0, 1, ..., 99\} \in \mathbb{R}^{100}$ is the age dictionary. We use Mean Absolute Error (Mean AE) and Median Absolute Error (Median AE) for evaluating the age estimation.

[4]Note that we did not implement the ResNet-like shortcuts, such as DenseNet. This is because that the ResNet-like shortcuts in DenseNet is less related to the NDDR layer we proposed. Therefore, we carefully stitch to the factors that influence the NDDR layer to analysis it more clearly.

	Age (Lower Better)		**Gender** (Higher Better)
	Mean AE	Median AE	Acc. (%)
Single-Task	9.1	7.4	83.5
Multi-Task	9.0	7.4	82.3
Cross-Stitch	8.6	7.0	**84.0**
Sluice	8.5	7.0	83.9
Ours	**8.0**	**6.2**	**84.0**

Table 7. Experimental results on age and gender classification.

The experimental results are show in Table 7. It shows that our method on age estimation significantly outperforms the state-of-the-art methods, *i.e.*, $(8.5 - 8.0)/8.5 \approx 5.9\%$ for Mean AE and $(7.0 - 6.2)/7.0 \approx 11.4\%$ for Median AE. While for the gender classification, our method just performs comparably with the cross-stitch network. This is because that gender classification is a two-class classification problem with sufficient labeled samples for each gender. Therefore, it benefits less from the other task (with another set of labels) when learning the representation.

6. Conclusions

In this paper, we proposed a novel CNN structure for general-purpose MTL. Firstly, the task-specific features with the same spatial resolution from different tasks were concatenated. Then, we performed Neural Discriminative Dimensionality Reduction (NDDR) over them to learn a discriminative feature embedding for each task, which also satisfies input sizes of the following layers.

The NDDR layer is simple and effective, which is constructed by combining existing CNN components in a novel way. The proposed NDDR networks can be trained in an end-to-end fashion without any extraordinary operations of a modern CNN. This desirable property guarantees that the proposed NDDR layer can easily be extended to various state-of-the-art CNN architectures in a "plug-and-play" manner. In addition, our proposed NDDR-CNN generalizes several state-of-the-art CNN based MTL models, such as the cross-stitch network [31] and the sluice network [40].

We performed detailed ablation analysis, showing that the proposed NDDR layer is easy to train and also robust to different hyperparameters. The experiments on various CNN structures and different task sets demonstrate the promising performance and desirable generalizability of our proposed method. An interesting future research direction can be studying explicitly imposing various dimensionality reduction assumptions on the NDDR layer.

Acknowledgments. This work is partially supported by NSFC 61773295, NSFC 61601112, ONR N00014-12-1-0883.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 2017. 1

[2] Liang-Chieh Chen, Jonathan T Barron, George Papandreou, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. In *CVPR*, pages 4545–4554, 2016. 1

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 1

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018. 1, 5, 6

[5] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016. 1

[6] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *CVPR*, pages 2650–2658, 2015. 2, 5

[7] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 2015. 6

[8] Ross Girshick. Fast R-CNN. In *ICCV*, pages 1440–1448, 2015. 1, 2

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014. 1, 2

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010. 6

[11] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *CVPR*, 2013. 5

[12] Hu Han, Anil K Jain, Shiguang Shan, and Xilin Chen. Heterogeneous face attribute estimation: A deep multi-task learning approach. *TPAMI*, 2018. 2

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1, 3, 5, 6

[15] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 1

[16] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 1, 3, 5, 7

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015. 2

[18] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018. 2

[19] Iasonas Kokkinos. Ubernet: Training auniversal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, 2017. 1, 2

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 6

[21] L'ubor Ladickỳ, Bernhard Zeisl, and Marc Pollefeys. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014. 5

[22] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. 2014. 3

[23] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, July 2017. 3

[24] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Deep multi-task learning with shared memory. In *EMNLP*, 2016. 2

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *ECCV*, pages 21–37. Springer, 2016. 1

[26] Wei Liu, Cun Mu, Rongrong Ji, Shiqian Ma, John R Smith, and Shih-Fu Chang. Low-rank similarity metric learning in high dimensions. In *AAAI*, 2015. 3

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 1

[28] Mingsheng Long and Jianmin Wang. Learning multiple tasks with deep relationship networks. In *NIPS*, 2017. 2

[29] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *CVPR*, 2017. 2

[30] A.M. Martinez and A. C. Kak. PCA versus LDA. *TPAMI*, 23(2):228–233, 2001. 3

[31] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016. 1, 2, 4, 5, 7, 8

[32] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 5, 7

[33] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *ICCV*, 2015. 1

[34] Rajeev Ranjan, Vishal M Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *TPAMI*, 2017. 2

[35] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *CVPR*, 2017. 1

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 1, 2

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015. 1

[38] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *IJCV*, July 2016. 7, 8

[39] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. 1

[40] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *AAAI*, 2019. 2, 5, 7, 8

[41] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 2

[42] Abhinav Shrivastava and Abhinav Gupta. Contextual Priming and Feedback for Faster R-CNN. In *ECCV*, 2016. 1, 2

[43] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training Region-based Object Detectors with Online Hard Example Mining. In *CVPR*, 2016. 1, 2

[44] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta. Beyond Skip Connections: Top-Down Modulation for Object Detection. *arXiv:1612.06851*, 2016. 1, 2

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 5, 6

[46] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. 3

[47] Peng Tang, Xinggang Wang, Song Bai, Wei Shen, Xiang Bai, Wenyu Liu, and Alan Loddon Yuille. Pcl: Proposal cluster learning for weakly supervised object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 1

[48] Peng Tang, Xinggang Wang, Xiang Bai, and Wenyu Liu. Multiple instance detection network with online instance classifier refinement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2843–2851, 2017. 1

[49] Ludovic Trottier, Philippe Giguère, and Brahim Chaib-draa. Multi-task learning by deep collaboration and application in facial landmark detection. *arXiv preprint arXiv:1711.00111*, 2017. 2

[50] Xiaolong Wang, Abhinav Shrivastava, and Abhinav Gupta. A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection. In *CVPR*, 2017. 1, 2

[51] Fangting Xia, Peng Wang, Liang-Chieh Chen, and Alan L Yuille. Zoom better to see clearer: Human part segmentation with auto zoom net. In *ECCV*, 2016. 1

[52] Fangting Xia, Peng Wang, Xianjie Chen, and Alan Yuille. Joint multi-person pose estimation and semantic part segmentation. In *CVPR*, 2017. 2

[53] Yanhua Yang, Cheng Deng, Shangqian Gao, Wei Liu, Dapeng Tao, and Xinbo Gao. Discriminative multi-instance multitask learning for 3d action recognition. *TMM*, 19(3):519–529, 2017. 2

[54] Yanhua Yang, Cheng Deng, Dapeng Tao, Shaoting Zhang, Wei Liu, and Xinbo Gao. Latent max-margin multitask learning with skelets for 3-d action recognition. *IEEE Transactions on Cybernetics*, 47(2):439–448, 2017. 2

[55] Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. In *ICLR*, 2017. 2

[56] Junho Yim, Heechul Jung, ByungIn Yoo, Changkyu Choi, Dusik Park, and Junmo Kim. Rotating your face using multi-task deep neural network. In *CVPR*, pages 676–684, 2015. 2

[57] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014. 1