Many Task Learning With Task Routing

Gjorgji Strezoski, Nanne van Noord and Marcel Worring

University of Amsterdam
{g.strezoski, n.j.e.vannoord, m.worring}@uva.nl

Abstract

Typical multi-task learning (MTL) methods rely on architectural adjustments and a large trainable parameter set to jointly optimize over several tasks. However, when the number of tasks increases so do the complexity of the architectural adjustments and resource requirements. In this paper, we introduce a method which applies a conditional feature-wise transformation over the convolutional activations that enables a model to successfully perform a large number of tasks. To distinguish from regular MTL, we introduce Many Task Learning (MaTL) as a special case of MTL where more than 20 tasks are performed by a single model. Our method dubbed Task Routing (TR) is encapsulated in a layer we call the Task Routing Layer (TRL), which applied in an MaTL scenario successfully fits hundreds of classification tasks in one model. We evaluate on 5 datasets and the Visual Decathlon (VD) challenge against strong baselines and state-of-the-art approaches.

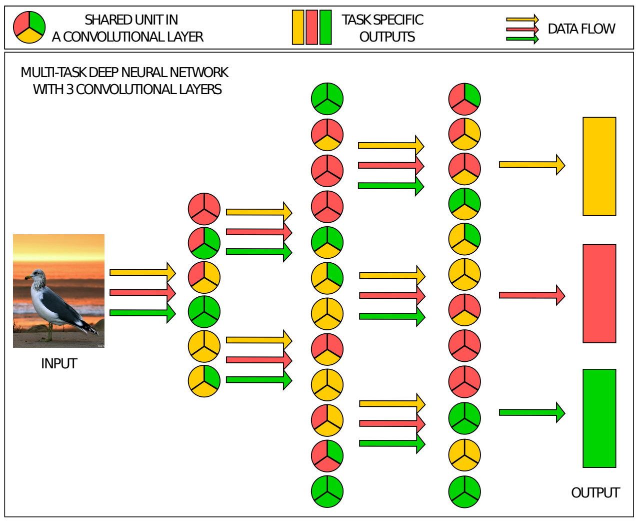
SHARED UNIT IN A CONVOLUTIONAL LAYER
TASK SPECIFIC OUTPUTS
DATA FLOW
MULTI-TASK DEEP NEURAL NETWORK WITH 3 CONVOLUTIONAL LAYERS
INPUT
OUTPUT

Figure 1: Routing map and specialized subnets through a three layer multi-task deep convolutional neural network with 50% of the units being shared per task routing layer.

1. Introduction

Multi-tasking is ubiquitous. In everyday life, as well as in computer science, performing multiple tasks at the same time improves efficiency and resource utilization [32, 39]. By definition, MTL is a learning paradigm that seeks to improve the generalization performance of machine learning models by optimizing for more than one task simultaneously [4]. Its counterpart, Single Task Learning (STL) occurs when a model is optimized for performing a single task only. STL models usually have an abundance of parameters that have the capacity to fit to more than one task [14]. In MTL the aim is to make use of this extra capacity. The simultaneously performed tasks can either help or hurt each others execution by sharing the expertise the model develops for each of them during training. For example, in a dataset of bird images, training a model to recognize white head feathers and white underbelly can help in classifying a Seagull bird. However, we cannot expect a bird size detector to help with Seagull classification as this bird appears in many sizes in nature, so size is not relevant to its species.

As with any combinatorial problem, in MTL there exists an optimal combination of tasks and shared resources which is unknown. Searching the space to find this combination is becoming increasingly inefficient, as modern models [6, 7, 28, 24] grow in depth, complexity and capacity. This search duration grows proportionally with the number of tasks and parameters present in the model's structure. Previous works in both MTL and STL rely on evolutionary algorithms [15] or factorization techniques [41] to discover their optimal way of learning, however this takes time and prolongs the training process. In our work, inspired by the efficiency of Random Search [3] we enforce a structured random solution to this problem by regulating the per-task data-flow in our models. As depicted in Figure 1, by assigning each unit to a subset of tasks that can use it we create specialized sub-networks for each task. In addition, we show that providing tasks with alternate routes through the parameter space, increases feature robustness and improves scalability while boosting predictive performance.

Creating and keeping alternate task routes throughout training, accounts for more than just learning a robust shared and task-specific feature representation. Distributing the per task knowledge regularizes the influence the tasks might have on each other, both positive and negative. Research in MTL has been implicitly attempting to solve the negative influence between jointly learned tasks. For example, statistically analyzing task relationships [1] and infusing the findings as prior knowledge helps combining tasks that benefit from each others learning process. In the same way, design choices for a complete multilevel output architecture with structured sharing such as Ubernet [10] can rest on firm domain experience. This allows [10] to create shared features between low, mid and high level tasks at the correct levels within the model. Prior knowledge can be utilized in the final branch-output of a branched MTL architecture as well, where carefully designing the task-specific branches improves the model's performance [23]. However, the above statistical analyses and architecture design choices rely on prior knowledge, which is often unavailable or expensive to obtain. We mitigate the issue of obtaining prior knowledge, by introducing a task-routing mechanism allowing tasks to have separate in-model data flows. In this way, by enforcing a structured random solution we allow tasks to forge their own beneficial way of sharing.

We empirically verify our routing mechanism's positive influence on the task number scalability capacity through gradually increasing the number of tasks performed by a single model in our experiments. Starting from a small number of tasks namely four in the Zappos-50K dataset, we scale up to 312 tasks in the UCSD-Birds dataset [36]. To distinguish this setup from regular MTL, we introduce Many Task Learning (MaTL) as a special case of MTL where more than 20 tasks are performed. For MTL we show competitive performance with a small task count, and further proceed beyond the capabilities of existing methods to achieve state-of-the-art performance on the complete set of possible tasks in the UCSD-Birds dataset in an MaTL context.

In this paper, we identify the following primary contributions:

- We present a scalable MTL technique for exploiting cross-task knowledge transferability without requiring prior domain knowledge.
- We enable structured deterministic sampling of multiple sub-architectures within a single MTL model.
- We define task relationships in an intuitive non-parametric manner during training without requiring prior domain knowledge or statistical analysis.
- We apply our method to a total of 15 datasets demonstrating its effectiveness and performance gains over strong baselines and state-of-the-art approaches.

2. Related Work

As our method draws inspiration from feature-wise transformation, architecture search and regularization works, this section is structured to cover those domains. As such, first we explain the ideas behind MTL and its possible variations. After that we link to related ideas in modulation and feature-wise transformations in an MTL context, and we complete the related work discussion by distinguishing our method from existing regularization and architecture search methods.

Multi-task learning (MTL) [2, 4, 34] is a learning paradigm which seeks to improve the generalization performance of machine learning models optimizing for more than one task. Caruana [4] further describes MTL as a mechanism for improving generalization properties by leveraging the domain-specific information contained in the training signals of related tasks. As such, in MTL the goal is to jointly perform experiments over multiple tasks and improve the learning process for each of them. Whether these experiments are optimized for simultaneously or in an incremental fashion, categorizes MTL approaches in either *symmetric* or *asymmetric* [39].

Asymmetric MTL relies on using knowledge from solving auxiliary tasks in order to improve the performance on one main target task. This formulation bears a resemblance to transfer learning [22]. One key distinction between them is that in asymmetric MTL the auxiliary tasks are learned simultaneously with the main task, while in transfer learning they are learned independently [45]. In our work we focus on symmetric MTL.

Symmetric MTL, unlike Asymmetric MTL, aims to improve the performance of all tasks simultaneously. It leverages the fact that some tasks are correlated (co-dependent) and by learning their estimators jointly under a unified representation, the transferability of expertise between tasks is exploited to the maximal benefit of all [39]. Zhang et al. introduced such a symmetric approach named Multi-task Relationship Learning (MTRL) [46] which regularizes the parallel learning of multiple tasks and models their relationships in a non-parametric manner as a task covariance matrix. Many other symmetric approaches [21, 44, 16, 13, 47, 43] have been developed in recent years. Combinations of utilizing different regularization strategies [13], multi-level sharing [43], cross-layer parameter combinations [21] or meshes of all options [26] have been exten-

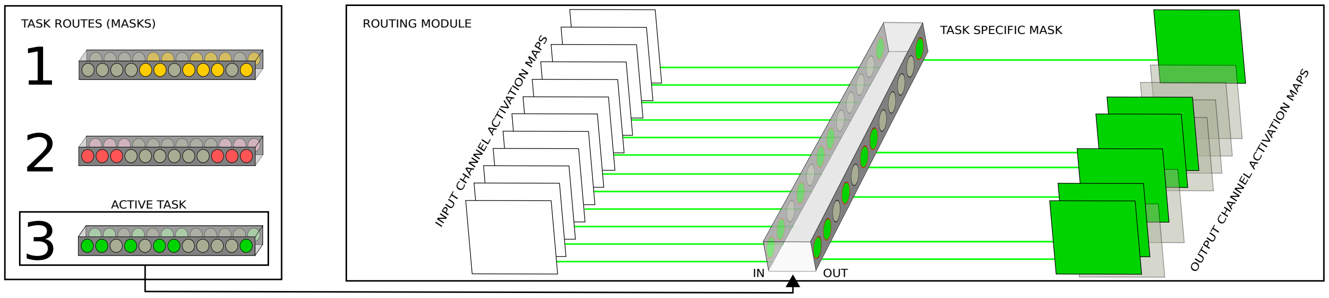

Figure 2: Operation of the TRL over the output from a convolutional layer (white channels). The current active task is used to select the mask (bright green are 1, and dark green are 0). After the element-wise multiplication across channels, the ones that remain are colored bright green and the nullified channels are brown and transparent.

sively tested. However, they are vulnerable to noisy and outlier tasks which when introduced dramatically deteriorate performance. This occurs due to the low grade feature robustness and the initial assumption that all tasks positively influence each other's learning process [45]. In our work we address the feature robustness issue by randomizing the sharing structure from the start of the training process and enforcing tasks to use alternate routes for their data-flow through the model.

Both symmetric and asymmetric MTL approaches often rely on prior knowledge to help with architecture design, sharing options and task grouping [23, 33, 9, 1]. If such knowledge is present, it is a helpful resource for designing an MTL model. However, often times such knowledge is unavailable and requires domain expert analysis (e.g. hand-crafting an MTL model for the Omniglot [12] dataset needs knowledge of ancient alphabets). For this reason developing MTL models without prior domain knowledge is crucial to real-world applications. A recent step in this direction is made by Liu et al.[18] who propose an adaptive MTL model that structurally groups tasks together. Evolutionary algorithms have also been shown to capture task relatedness and create sharing structures [15]. A less architectural solution is proposed by Yang et al. [41] who use a factorized space representation to learn inter-task sharing structures at each layer in an MTL model. Most of these methods have firm constraints in terms of how a model should be defined, structured, or initialized. We propose an approach applicable to any deep MTL model with no structural adjustments, as we encapsulate the layer-wise parameter space. By controlling the data-flow instead of the structure, we do not affect the underlying model behavior and this broadens the usability scope of our method.

Resource consumption is becoming predominantly important in MTL models as it usually increases with the number of performed tasks. As our approach is not structure dependent, it has a very small computational footprint. A recent scalable approach to MTL using modulation for image retrieval was proposed in [48] where they successfully scale to performing 40 tasks. The trade-off between speed and memory size in [48] shows only 15% overhead. In our work we build on this approach and demonstrate competitive performance on 300+ tasks in a single model with minimal costs to the computational budget, which with current methods is either inefficient or impossible.

PackNet [20] presents an idea related to our work in the sense that Mallya et al. use the fixed weights of an existing network to learn a new task with the same model. This is an intuitive and simple method for re-usability of backbone networks for performing additional tasks, however as the authors indicate it has the downside of not allowing the tasks to share and benefit from each other's learning process. In cases such as this, Adaptive Instance Normalization [8] is an approach that is able to adapt the channel-wise mean and variances between two inputs with no learnable parameters. This offers a similar feature-wise transformation and does not suffer from the same issue as [20], but has not been tested in an MTL scenario where the inputs are task specific representations.

Convolutional neural fabrics [27] is related to our work in terms of architecture search. Saxena et al. define 3D trellis that connect response maps from different layers and create a smaller, thinner specialized architecture. On the other hand, TR works in the MTL realm and allows us to pool from an exponentially large pool of subnetworks.

Similarly, Dropout [31] relates to our work as a form of regularization and co-adaptation prevention technique. In Dropout, the dropped-out units change each time the Bernoulli vector is sampled, which adds a stochastic component to this technique further inhibiting unit co-adaptation. The same regularization building block is present in related approaches [30, 37, 5, 40] in a STL scenario. TR allows for a more deterministic form of inter-task regularization in a symmetric MTL paradigm. Furthermore, Dropout can be applied and prove beneficial in combination with our method, as it can provide general regularization and additional co-adaptation prevention during training.

3. Task Routing

Most MTL methods involve task specific and shared units as part of the MTL training procedure. Our method enables the units within the model's convolutional layers to have a consistent *shared* or *task-specific* role in both training and testing regimes. Figure 1 provides the intuition behind how the individual units are utilized throughout the set of tasks performed by the model. We achieve this behavior by applying a channel-wise task-specific binary mask over the convolutional activations, restricting the input to the following layer to contain only activations assigned to the task. Figure 2 illustrates the masking process over the activations. Because the flow of activations does not follow its conventional route, i.e. it has been rerouted to an alternate one, we have named our method Task Routing (TR) and its corresponding layer the Task Routing Layer (TRL). By applying the TRL to the network we are able to reuse units between tasks and scale up the number of tasks that can be performed with a single model.

The masks that enable task routing are generated randomly at the moment the model is instantiated and kept constant through the training process. These masks are created using a sharing ratio hyper-parameter σ defined beforehand. The sharing ratio defines how many units are task specific, and how many are shared between tasks. The inverse of this ratio determines how many of the units are nullified. As such, the sharing ratio enables us to explore the complete space of sharing possibilities with a simple adjustment of one hyper-parameter. A sharing ratio of 0 would indicate that no sharing occurs within the network and each trainable unit is specific to a single task only, resulting in distinct networks per task. On the opposite side of the spectrum, a sharing ratio of 1 would make every unit shared between each of the tasks, resulting in a classical fully shared MTL architecture.

3.1. Task Routing Mask Creation

Task routing is performed by means of a feature-wise transformation of the unit activations in a convolutional layer with a conditional binary mask. As our system does not have any prior knowledge for the problem at hand, the masks are created randomly at the moment our model is instantiated. The resulting random structure is persistent through the training and testing periods as the masks are not trainable.

Having immutable masks is particularly useful for MaTL, in which the space of possible sharing strategies is very large. By enforcing a fixed sharing strategy from the start of the training process, the model can focus on training robust task-specific and shared units, as opposed to training units over an ever changing combination of tasks.

3.2. Task Routing Layer

We propose a new layer dubbed *Task Routing Layer (TRL)* which contains a task specific binary mask $m_A \in \mathcal{Z}_2^C$ for the active task A applied over the input $X \in \mathcal{R}^{C\times H\times W}$ of a convolutional layer with dimensionality $[B \times C \times H \times W]$ where B is the batch size, C is the number of units, H the height and W the width of the unit. For simplicity of notation we drop the H and W dimensions, as the mask is applied to an entire channel uniformly across the spatial dimensions. Applying this mask (see equation 1) is akin to performing a conditional feature-wise transformation and generates a masked output which is then propagated to the next convolutional block. Figure 3 shows the TRL placement within a convolutional block. Because the feature-wise transform applied to the convolutional output can affect the local running mean and variance, the TRL is placed right after the batch normalization layer (if present).

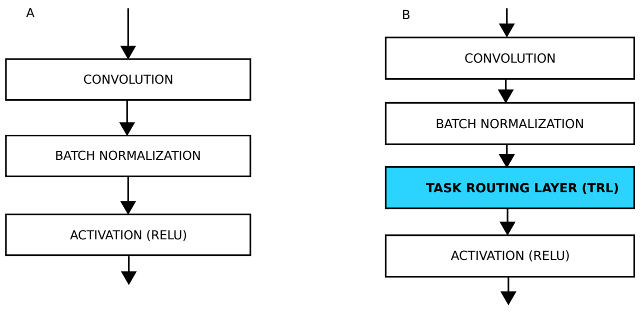

Figure 3: TRL placement (blue block) within a convolutional block. Section A (on the left) shows the convolutional block before adding the TRL, and Section B after.

$$TRL_A(X) = m_A \odot X \tag{1}$$

During a forward pass a single specialized subnet is active, namely the subnet for the active task A. This is achieved by setting the active task for the TRLs. During our forward propagation we randomly sample one task from the pool of tasks. As the number of iterations required to traverse the dataset is usually much higher than the number of tasks, there exists a very small chance that a task is not optimized for within an epoch. This chance diminishes drastically with the training process spanning over multiple epochs. Even if we consider that a task has not been optimized for in an epoch, this is easily compensated for by the optimization of the other tasks that partly share the same group of units. At test time, a separate per task evaluation is required as the input has to propagate distinct subnets for each task.

The training operational flow of our method is illustrated in algorithm 1. As we iterate through the training set, with each sampled mini-batch we change the currently active

Algorithm 1 Training epoch for TRL

```
1: procedure TRAIN(X)
2:     for X in $X_{Train}$ do                ▷ Training loop
3:         $A \leftarrow sample(task\_set)$
4:         set_active_task($A$)
5:         forward($X$)
```

task. Setting the currently active task is a global change in the framework, so the TR work-flow does not affect existing ways of propagation and training. This property makes TR simple to integrate in existing projects. As a global variable, the active task affects the applied mask in all TRLs in the model and navigates the routed activations to the task specific classifier.

In the forward pass of the input, the TRL applies the selected mask for the active task uniformly across the spatial dimensions of the entire input batch. As demonstrated in algorithm 2 we perform the feature-wise transformation in the forward pass of the TRL over the convolutional layer's output. Figure 2 illustrates how the task routing is performed and how the channels are nullified by the active mask.

Algorithm 2 Forward pass for TRL

```
1: procedure FORWARD(X)
2:     $m_A \leftarrow M[A]$            ▷ $M \leftarrow setofmasks$
3:     $out \leftarrow m_A \odot X$       ▷ Across all channels
4:     return $out$                ▷ The masked output
```

3.3. Complexity

Our model only adds a minimal number of additional parameters compared to the hard shared MTL approaches [4] or cross-stitch networks [21], and has a significantly lower parameter count compared to similar architecture search approaches [27]. The models we define in our experimental setup contain the task routing layers after each convolutional layer. This way, the number of additional parameters of the TRL is directly linked and proportional to the number of convolutional layers, units and channels.

Increasing the number of tasks, without appending a distinct embedding per task results in an negligible parameter count increase. However, heuristically we determined that having a separate embedding space per task increases standalone task performance. Because of this, the majority of the additional parameters in our models come from the wide task specific branches, rather than the TRLs.

4. Experimental Design

Our experiments are designed to test and validate the contributions presented in this work. We evaluate our approach on multiple classification tasks, comparing to strong baselines and state of the art approaches. For this we consider a variety of datasets ranging from grayscale proof of concept datasets (FashionMNIST), to attribute rich real world problems (UCSD-Birds), cross-dataset MTL benchmarks (VD) and a multi-attribute based face dataset (CelebA). In addition, through the CelebA and UT-Zappos50K dataset we compare to the state of the art performance presented in [48].

4.1. Datasets

UCSD Birds [36] is a dataset that provides 11.788 bird images over 200 bird species with 312 binary attribute annotations. For state of the art comparison, we compare on ten target attributes obtained with spectral clustering using the FSIC as the similarity measure [1]. As we incrementally increase the selected number of attributes we define sets of 50, 100, 200, and 312 binary classification tasks for each of them. For this dataset the training and testing set have equal sizes and distributions. The attributes are sampled according to the ten attribute selection in [1] for the 10 task experiment and in order of the original annotation file for the rest of the experiments.

Visual Decathlon (VD) [25] is a benchmark that evaluates the ability of representations to capture simultaneously ten very different visual domains and measures their ability to perform well uniformly. While the images of this task are of a lower resolution (72x72 px.), they contain a wide variety of tasks such as pedestrian, digit, aircraft, and action classification, making it perfect for testing the generalization abilities of our method. We evaluate our performance with per-task accuracies, and assign a cumulative score with a maximum value of 10,000 (1,000 per task) based on the per-task accuracies using the official challenge metric [19].

FashionMNIST [38] and **CIFAR-10** [11] constitute the proof of concept part of our experimental design as they are well established benchmarks and provide indication of how different hyper-parameter setups tend to affect the method. For both datasets we define ten binary classification tasks and evaluate the accuracy, precision, and recall scores.

CelebA [17] consists of more than 200,000 face images with binary annotations on 40 facial attributes. The first 10 (out of 40) attributes from [48] are selected for the 10 task experiment as more related to face appearance. We additionally report the results on 40 attributes to compare our approach to [48] in a classification setting.

UT-Zappos50K [42] is a large shoe dataset consisting of more than 50,000 catalog images collected from the web. This dataset contains four attributes of interest for our experiment, namely shoe type, suggested gender, height of the heel, and the shoe closing mechanism. As defined in [48], we define 4 classification tasks for a small scale test of our approach over a real world dataset using the identical train, validation, and test splits from [35, 48].

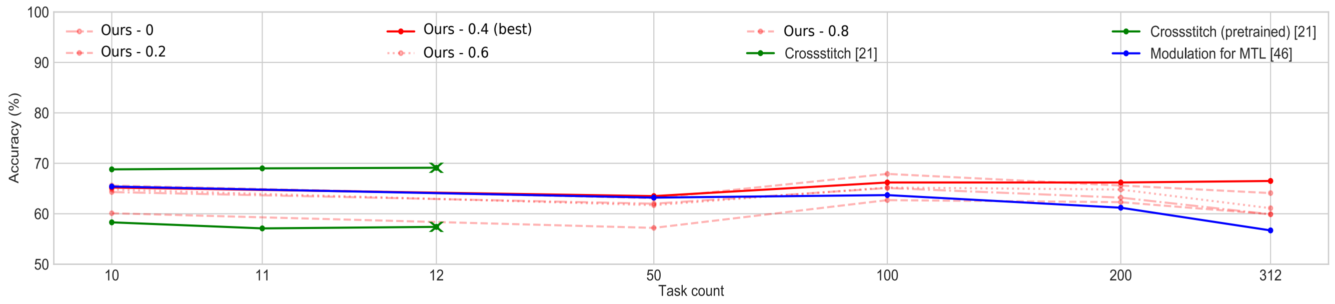

Figure 4: Accuracy comparison on the UCSD-Birds dataset on 10, 50, 100, 200, 312 task between our method (in red) with $\sigma = [0, 1]$, cross-stitch networks [21] (in green) and modulation for MTL [48] (in blue). Cross-stitch networks scale up to 12 tasks, where as modulation for MTL and our approach fit to the full number of tasks. The best performing sharing ratio $\sigma = 0.4$ is set in strong red, and the other σ values in light red.

Table 1: Average scores on the VD challenge. Best overall approach is highlighted in gray.

Run	VD Score	Aircraft	Cifar-100	Daimler	DTD	GTSRB	ImageNet-12	Omniglot	SVHN	UCF-101	VGG-Flowers
ResAdapt [25] ($\sigma = 0$)	2851.31	299.88	195.96	155.41	261.51	472.6	224.15	337.05	**282.8**	231.69	390.26
Ours $\sigma = 0.2$	2873.84	302.1	200.01	162.79	267.22	472.2	210.2	344.12	265.4	250.02	399.78
Ours $\sigma = 0.4$	**2919.26**	**305.2**	204.12	**165.89**	**273.28**	469.2	**228.39**	345.08	272.77	252.12	**403.21**
Ours $\sigma = 0.6$	2870.26	287.2	206.12	148.89	256.28	474.2	223.39	**350.08**	260.77	261.12	402.21
Ours $\sigma = 0.8$	2806.26	285.2	208.12	139.89	253.28	455.2	222.39	338.08	249.77	**263.12**	391.21
Ours $\sigma = 1$	2768.26	282.2	**214.12**	132.89	256.28	445.2	207.39	339.08	239.77	261.12	390.21

4.2. Multi-task Setup

FashionMNIST and CIFAR10 power a toy problem experiment where we develop an intuition of how our method functions. We choose these datasets as they are well established and balanced benchmarks, for which little domain knowledge is necessary to interpret the results and draw conclusions. For each dataset we perform 10 binary classification tasks using the model presented by Xiao et al. [38] for FashionMNIST and a VGG-16 network for CIFAR-10. In a similar manner, the Zappos50K dataset provides a highly related balanced dataset with four well described tasks on which we evaluate our method's performance in a small scale MTL context.

To evaluate our method in an MaTL context, we perform experiments on the CelebA and UCSD-Birds datasets. For the CelebA experiment we ran experiments with an increasing number of tasks, starting from 10 and ending with 40 tasks. The purpose of these experiments is to observe the difference in performance with [48] and explore how adding additional tasks affects the learning process and performance. This CelebA experiment uses the VGG-16 model trained from scratch with batch normalization as a feature extraction platform and branches out to as many classification branches as there are tasks. Each classification branch is task specific and has an independent embedding space. With 312 possible attributes related to bird appearance, the UCSD-Birds dataset presents a unique opportunity to explore the task-wise scalability properties of our method. To ensure a fair comparison with cross-stitch networks [21] we evaluate this method in a pre-trained and trained from scratch setting where applicable.

For the VD challenge we compare to the best approach from the leader-board [25] and use their Residual Adapters model fitted with TRLs over the complete range of $\sigma = [0, 1]$. The initialization and hyperparameter setup is the same as in the original paper [25].

4.3. Implementation Details

In all classification settings we perform binary classification tasks over the attributes. Each attribute is considered a binary classification task and has its own equal-sized embedding space. We append the TRL after each convolutional layer in our models and randomly initialize the routing maps as the model is instantiated. For all our experiments we use existing model architectures (VGG-11, VGG-16 [29] and Resnet50 [6]) with their default settings. For all datasets we use a batch size of 64 images with normalization by the dataset mean. We perform the VD challenge with the same settings and hyper-parameters from [25]. For the UCSD-Birds dataset we explore both trained from scratch and pretrained VGG-11 models with horizontal flipping because of the small size of the training set. We use Stochastic Gradient Descent with a learning rate of 0.01 and momentum of 0.5 with which our method usually converges after 35 epochs [1].

[1]Code available at: https://github.com/gstrezoski/TaskRouting

Table 2: Average scores over 5 runs on the on FashionMNIST, CIFAR-10, Zappos50K and CelebA (10 and 40 Tasks) on the complete dataset with the complete sharing ratio scope $\sigma = [0, 1]$. Because for $\sigma = 0$ no sharing occurs and for $\sigma = 1$ our approach reverts to hard shared MTL, we group them separately. The overall best performing approach across all datasets is highlighted in gray and the best approaches per dataset are in bold. Fields marked with *n/a* signify experiments for which the method could not scale to the task count.

Dataset	FashionMNIST			CIFAR-10			Zappos50K			CelebA			CelebA (Full)		
Number of Tasks	10			10			4			10			40		
Approach	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Cross-stitch	**98.1**±1.14	91.4±1.02	86.1±0.23	98.5±0.17	91.6±1.18	85.9±1.07	84.7±2.23	82.2±1.12	81.8±1.29	71.5±1.96	68.0±1.38	67.0±0.83	n/a	n/a	n/a
Modulation	96.9±2.04	91.0±1.14	80.1±0.54	63.2±1.13	57.4±2.14	53.2±3.10	63.7±2.76	60.4±1.94	59.8±2.02	71.9±1.66	70.2±2.18	69.4±2.63	64.1±1.43	61.0±1.81	60.4±1.45
Ours σ = Adapt	96.3±0.04	90.6±0.04	84.1±0.05	98.1±0.06	88.3±0.03	85.9±0.05	88.3±0.11	81.7±0.02	80.6±0.04	71.9±0.07	68.2±0.08	66.3±0.17	63.0±0.08	59.0±0.15	57.1±0.11
Ours $\sigma = 0$	97.8±0.25	91.9±0.44	85.5±0.32	96.5±0.42	89.1±0.98	**87.8**±0.24	88.3±0.31	83.1±0.54	83.2±0.42	70.1±0.08	68.0±0.22	67.4±0.78	63.1±0.33	60.8±0.05	60.0±0.21
Ours $\sigma = 1$	97.4±0.01	91.1±0.07	85.1±0.04	98.1±0.03	88.0±0.03	85.6±0.01	79.2±0.10	77.1±0.09	75.3±0.08	69.9±0.13	67.2±0.10	66.8±0.06	62.2±0.07	58.0±0.07	56.4±0.11
Ours $\sigma = 0.2$	97.8±0.06	**92.2**±0.11	**85.7**±0.03	**99.0**±0.03	**92.1**±0.08	85.2±0.06	**89.5**±0.03	**85.2**±0.04	**83.4**±0.12	**73.2**±0.15	**71.4**±0.11	**70.8**±0.13	63.1±0.12	60.8±0.12	60.0±0.13
Ours $\sigma = 0.4$	97.6±0.05	92.0±0.08	84.2±0.07	96.9±0.09	92.0±0.09	87.5±0.15	88.1±0.17	84.3±0.18	82.8±0.14	73.0±0.14	71.4±0.12	70.2±0.12	62.0±0.25	59.4±0.24	58.2±0.24
Ours $\sigma = 0.6$	97.1±0.10	91.1±0.08	80.4±0.08	96.0±0.06	90.3±0.05	84.3±0.07	87.4±0.11	82.2±0.17	82.0±0.13	72.7±0.05	71.0±0.04	69.6±0.09	**65.3**±0.22	**62.4**±0.18	**61.8**±0.17
Ours $\sigma = 0.8$	96.8±0.08	91.0±0.08	78.0±0.03	94.8±0.09	88.2±0.11	81.1±0.10	83.2±0.04	81.4±0.01	78.9±0.01	71.4±0.05	70.1±0.06	69.1±0.08	64.0±0.14	61.1±0.10	60.2±0.09

4.4. Evaluation criteria

For evaluating the performance of our method we track the accuracy, precision and recall. We added precision to our evaluation criteria because it is important to highlight how precise the model is, i.e. how many of the positive predictions are true positives. This gives insight into how precise and how robust our specialized subnets are the task specific representation is. Tracking recall gives a realistic measure how well the model has adapted to each of the tasks as it shows the coverage of actual positive samples.

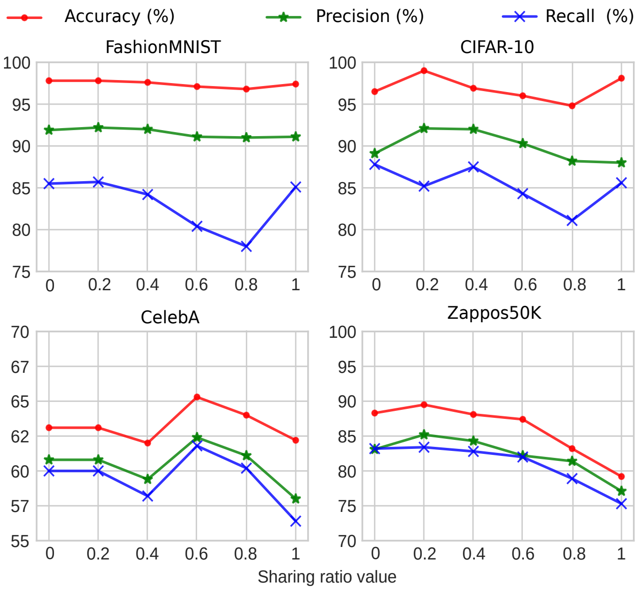

Figure 5: Effects of the sharing ratio σ value on accuracy, precision and recall in Fashion-MNIST (top left), CIFAR10 (top right), CelebA (bottom left) and Zappos50K (bottom right). Partially sharing units between tasks is beneficial to performance with sharing ratios $\sigma = [0.2, 0.6]$ compared to a fully shared network $\sigma = 1$ or many distinct subnetworks without sharing $\sigma = 0$.

5. Results

We evaluate our method on five datasets with experiments ranging from proof of concept (FashionMNIST, CIFAR10 and Zappos50K) to addressing the task count scalability properties (UCSD-Birds and CelebA) and a complete range of vision tasks (VD). We report performance for the full scope of possible values for our method specific hyperparameter σ, as compared to modulation for MTL [48] and cross-stitch networks [21]. With $\sigma = [0, 1]$ and a varying number of tasks per model we explore the complete space of sharing capabilities our method has to offer ranging from a distinct specialized subnet per task $\sigma = 0$, to a completely shared structure when $\sigma = 1$.

The experimental results for this comparison are reported in Tables 1, 2 and3. The results show that using our method we are able to efficiently use the units in a single model to fit and optimize for many tasks. Moreover, we surpass the performance of the modulation for MTL method [48], as well as the cross-stitch networks approach [21] over five datasets in an MTL/MaTL setup (Figure 4). In the VD challenge, by adding the TRL to [25] we report a boost in performance on 9 out of 10 datasets (see Table 1).

An important feature of our method is its scalability regarding the number of tasks a single model can accommodate using the TRL. Table 3 shows the relationship between the number of tasks, the sharing ratio coefficient σ and accuracy, precision and recall as evaluation metrics. For sharing ratios $\sigma = 0.2$ and $\sigma = 0.4$ we can observe a consistent improvement on all three scores as the task count is increased in Table 3. Performance significantly drops with a sharing ratio of $\sigma = 0.8$ as the model is closing in on a hard shared MTL architecture. In this case only 20% of the units remain task specific after the routing. A sharing ratio of $\sigma = 1$ means that every unit is used by every task and the reported performance is equal to that of a classical MTL hard shared approach (see Table 2 and 3). Figure 5 shows the effects of the sharing ratio parameter σ over the evaluation met-

Table 3: Average scores using the routing module over an increasing number tasks for the UCSD-Birds dataset and a sharing ratio of $\sigma = [0, 1]$. Because for $\sigma = 0$ no sharing occurs and for $\sigma = 1$ our approach reverts to hard shared MTL, we group them separately. Fields marked with *n/a* signify experiments for which the method could not scale to the task count. The pretrained cross-stitch networks experiment is marked with a star (*). The overall best performing method is highlighted in gray and the best performing model per task setting is set in bold.

Dataset	UCSD-Birds														
Number of tasks	10			50			100			200			312		
Approach	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Cross-stitch [21]	58.3	55.6	54.2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Cross-stitch [21] *	**68.8**	**67.4**	**67.0**	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Modulation [48]	65.4	59.8	55.2	63.2	57.4	53.2	63.7	60.4	59.8	61.2	58.6	57.3	56.7	51.8	50.2
Ours $\sigma = 0$	64.3	62.4	55.3	62.0	60.6	54.6	65.1	62.7	61.1	63.2	60.2	58.8	59.9	57.2	56.1
Ours $\sigma = 1$	62.3	57.4	51.8	58.6	56.8	54.2	60.7	58.1	57.8	60.0	58.6	56.8	59.6	53.9	52.2
Ours $\sigma = 0.2$	65.6	62.9	57.0	63.1	62.9	57.2	**67.8**	63.3	60.9	65.6	63.6	63.2	64.1	61.6	60.2
Ours $\sigma = 0.4$	65.1	62.7	55.9	**63.5**	**63.0**	**59.9**	66.2	**63.8**	**61.2**	**66.2**	**64.2**	**63.7**	**66.5**	**62.3**	**61.8**
Ours $\sigma = 0.6$	64.9	62.1	54.8	61.7	59.9	59.0	65.2	60.9	59.5	64.8	62.0	59.8	61.1	59.2	59.0
Ours $\sigma = 0.8$	60.1	55.0	50.2	57.2	52.2	50.0	62.7	60.4	59.8	62.3	59.2	58.0	59.9	55.1	54.2

rics. For $\sigma = 0$ the model converts to a hard shared solution, and for $\sigma = 1$ we have vanilla soft sharing. For simple problems, the discriminatory filters are often low level. Sharing the complete layer, or not sharing at all is expected to yield similar results. However, for the internal range ($\sigma = [0.1, 0.9]$) where performance peaks the behavior is different. Lower σ values allow for more task specific filters which benefits complex tasks where fine-grained details are key (UCSD-Birds). The lower σ range proves beneficial for cross-dataset scenarios as well (VD challenge) where we obtain best performance with $\sigma = 0.2$ and $\sigma = 0.4$.

For the CelebA we performed an experiment with ten tasks and 40 tasks (the complete attribute set). We consistently witness a performance drop across all applicable approaches once the remaining 30 tasks are added to the task pool. From these results, a plausible conclusion is that it is more difficult to perform well on the additional 30 tasks. We suspect that this is due to a smaller sample number of positive instances in the training set for the additional tasks when used in a classification setting.

Considering the task count scalability of our method compared to competing approaches, Figure 4 illustrates the task count to performance relation for cross-stitch networks [27], modulation for MTL [48], a hard shared MTL baseline (our approach with $\sigma = 1$) and our approach over the complete sharing space ($\sigma = [0, 0.8]$). As cross-stitch networks require independent models per task between which unit sharing occurs, every additional task requires a complete model to be loaded into memory. Despite having good performance even with using a small VGG-11 network, it becomes impossible to fit this approach into memory with more than 12 tasks. On the other hand, our approach and [48] are more parameter savvy and can fit more tasks. However, when comparing the performance of our approach to modulation for MTL we can observe a slight gain in performance for our approach as task count increases, whereas modulation for MTL drops in performance.

6. Conclusion

In this work, we have presented a method to efficiently perform a large number of classification tasks with a single model. The proposed method allows us to modify the default behavior of an MTL model and apply a conditional feature-wise transformation to the outputs of its convolutional layers. A strong point of our approach is that it does not require prior knowledge of the domain or the intertask relationships to achieve good performance in both regular MTL and MaTL settings.

At the core of our method is a layer dubbed the *Task Routing Layer*, that can be inserted after any convolutional layer within a model's architecture with minimal effort and computational overhead. This layer contains task specific masks that allow for a single model to fit to many tasks within its parameter space. By passing the input through the mask we are training specialized subnetworks per task with much lower dimensionality compared to the main model. The dimensionality of the specialized subnets is dictated by the sharing ratio hyper-parameter, and they can be extracted and used in place of the complete model for a specific task.

Furthermore, the sharing ratio hyper-parameter σ gives our method an additional degree of freedom when compared to other state-of-the-art approaches and baseline methods. The sharing ratio σ allows us to be flexible with the task routing design without changing the underlying architecture which proves beneficial in MTL and MaTL. As an universal solution to most of the problems does not exist, we offer a simple way to explore all sharing possibilities the model has to offer. Finally, our approach offers an intuitive and easy to implement mechanism to get more out of existing models.

Acknowledgments The authors would like to thank Pascal Mettes, William Thong and Devanshu Arya for their feedback and discussions. This research is supported by the VISTORY project NWO award number 628.007.004.

References

[1] Youssef Alami Mejjati, Darren Cosker, and Kwang In Kim. Multi-task learning by maximizing statistical dependence. In *Proceedings of CVPR*, 2 2018.

[2] Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4(May):83–99, 2003.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

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

[5] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pages 10750–10760, 2018.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[8] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[9] Brendan Jou and Shih-Fu Chang. Deep cross residual learning for multitask visual recognition. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 998–1007. ACM, 2016.

[10] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6129–6138, 2017.

[11] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[12] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[13] Sijin Li, Zhi-Qiang Liu, and Antoni B Chan. Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 482–489, 2014.

[14] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8168–8177, 2018.

[15] Jason Liang, Elliot Meyerson, and Risto Miikkulainen. Evolutionary architecture search for deep multitask networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 466–473, New York, NY, USA, 2018. ACM.

[16] Wu Liu, Tao Mei, Yongdong Zhang, Cherry Che, and Jiebo Luo. Multi-task deep visual-semantic embedding for video thumbnail selection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3707–3715, 2015.

[17] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset.

[18] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[19] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[20] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[21] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

[22] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[23] 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. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):121–135, 2019.

[24] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*, 2018.

[25] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516, 2017.

[26] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent Multi-task Architecture Learning. *arXiv e-prints*, May 2017.

[27] Shreyas Saxena and Jakob Verbeek. Convolutional neural fabrics. In *Advances in Neural Information Processing Systems*, pages 4053–4061, 2016.

[28] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] Saurabh Singh, Derek Hoiem, and David Forsyth. Swapout: Learning an ensemble of deep architectures. In *Advances in neural information processing systems*, pages 28–36, 2016.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way

to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[32] Ewa Szumowska, Małgorzata Kossowska, and Arne Roets. Motivation to comply with task rules and multitasking performance: The role of need for cognitive closure and goal importance. *Motivation and emotion*, 42(3):360–376, 2018.

[33] Marvin Teichmann, Michael Weber, Marius Zoellner, Roberto Cipolla, and Raquel Urtasun. Multinet: Real-time joint semantic reasoning for autonomous driving. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1013–1020. IEEE, 2018.

[34] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pages 640–646, 1996.

[35] Andreas Veit, Serge Belongie, and Theofanis Karaletsos. Conditional similarity networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 830–838, 2017.

[36] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[37] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.

[38] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[39] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8(Jan):35–63, 2007.

[40] Yoshihiro Yamada, Masakazu Iwamura, and Koichi Kise. Shakedrop regularization. *arXiv preprint arXiv:1802.02375*, 2018.

[41] Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. In *Proceedings of the 2017 International Conference on Learning Representations*, 2017.

[42] A. Yu and K. Grauman. Semantic jitter: Dense supervision for visual comparisons via synthetic images. In *International Conference on Computer Vision (ICCV)*, Oct 2017.

[43] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018.

[44] Wenlu Zhang, Rongjian Li, Tao Zeng, Qian Sun, Sudhir Kumar, Jieping Ye, and Shuiwang Ji. Deep model based transfer and multi-task learning for biological image analysis. *IEEE transactions on Big Data*, 2016.

[45] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

[46] Yu Zhang and Dit-Yan Yeung. A regularization approach to learning task relationships in multitask learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(3):12, 2014.

[47] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision*, pages 94–108. Springer, 2014.

[48] Xiangyun Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. A modulation module for multi-task learning with applications in image retrieval. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 415–432, Cham, 2018. Springer International Publishing.