CompositeTasking: Understanding Images by Spatial Composition of Tasks

Nikola Popovic[1], Danda Pani Paudel[1], Thomas Probst[1], Guolei Sun[1], Luc Van Gool[1,2]
[1]Computer Vision Laboratory, ETH Zurich, Switzerland
[2]VISICS, ESAT/PSI, KU Leuven, Belgium
{nipopovic, paudel, probstt, guolei.sun, vangool}@vision.ee.ethz.ch

Abstract

We define the concept of CompositeTasking as the fusion of multiple, spatially distributed tasks, for various aspects of image understanding. Learning to perform spatially distributed tasks is motivated by the frequent availability of only sparse labels across tasks, and the desire for a compact multi-tasking network. To facilitate CompositeTasking, we introduce a novel task conditioning model – a single encoder-decoder network that performs multiple, spatially varying tasks at once. The proposed network takes an image and a set of pixel-wise dense task requests as inputs, and performs the requested prediction task for each pixel. Moreover, we also learn the composition of tasks that needs to be performed according to some CompositeTasking rules, which includes the decision of where to apply which task. It not only offers us a compact network for multi-tasking, but also allows for task-editing. Another strength of the proposed method is demonstrated by only having to supply sparse supervision per task. The obtained results are on par with our baselines that use dense supervision and a multi-headed multi-tasking design. The source code will be made publicly available at www.github.com/nikola3794/composite-tasking.

1. Introduction

Intuitively, different image understanding tasks offer complementary information for scene understanding and reasoning [24, 53, 2, 62, 57, 61, 11, 50]. Therefore, networks that can perform multiple visual tasks on the same image are of very high interest [10, 34, 22, 51, 55]. A key aspect – effectively serving the ultimate goal of scene understanding and reasoning – is often not part of their design. This paper is about this utility question: can we determine *where in the image it is necessary (or even meaningful) to perform a task?* For example, the task of recognizing human body parts is meaningful only in the presence of humans. Similarly, any attempt to estimate the normals of the sky is absurd.

One may argue that we cannot know beforehand whether some task is necessary to be performed, without recognizing the image content. The content of the image may then reveal the task necessity. This begs the question whether we can know what task needs to be performed where, while bypassing the content-task pairing altogether? When the answer to *where* is known – either by learning or not – we aim to design an algorithm that executes the given multi-task instructions in an efficient manner. For example, some applications of Augmented Reality may require human poses and the normals of the interacting surfaces. We show that such flexibility to locally activate some tasks allows us to design more compact multi-tasking networks.

The task specific annotations of images are often sparse, either by definition or due to missing annotations. Take, for example, facial landmarks or image salience. Sometimes, the annotations may be missing simply because of being futile. Even the well-curated PASCAL-MT [38, 34] dataset has the sparsity of 30.4%, 7.5%, 41.9%, 60.0%, for semantic segmentation, human body parts, surface normals, and salience, respectively. Such label sparsity only tends to get worse, if the image annotations are crowd-sourced. In fact, it is simply impractical to expect pixel-wise dense annotations for large datasets, even at locations where the annotations are well defined. The case of merging datasets, by cross-label intersection, follows the same behaviour of sparsity. Under such circumstances, it may be unnecessary to waste computational resources during learning, for image pixels without labels. This calls for an efficient learning paradigm for multi-tasking from sparse labels. In this work, we show that efficient learning from sparse multi-task labels and executing spatially chosen multi-task instructions go hand-in-hand.

The key idea of this paper is rather simple. We design a convolutional neural network that performs multiple, pixel-wise tasks. We feed every image along with a composition of spatially distributed multiple task requests – which we call Task Palette – to execute pixel-specific tasks. This process we call CompositeTasking. The proposed network uses a single encoder-decoder architecture to perform all

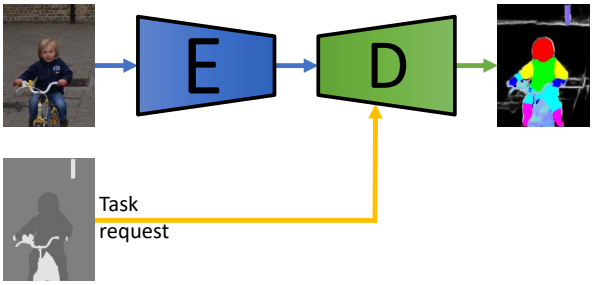

Figure 1: **CompositeTasking.** Given an RGB image and a Task Palette as inputs, our CompositeTasking performs locally request-specific tasks to compute the output.

the tasks in one forward pass. The simplicity of such architecture allows us to perform multiple tasks in an efficient and compact manner, thanks to the proposed method. An overview of our network is presented in Fig. 1.

The proposed method for CompositeTasking learns by task-specific batch normalization. Each task is performed by predicting layer-wise (only on the decoder side) affine batch normalization (BN) parameters, using a small task-conditioned network. Such design choice dedicates the encoder towards a compact visual representation shared among tasks. On the output side, each task is represented in an image format – thereby performing the conditional image-to-image translation. The image format chooses some arbitrary embedding for every task. We aim to map images to such embedding, conditioned upon the spatially distributed task requests. The task specific losses are then computed by mapping the predicted embedding to the task-appropriate label representations. For example, the predicted 3-channel values are mapped to class probabilities for segmentation task to compute cross-entropy, whereas, pixel normals are directly regressed by minimizing the angular distance between the prediction and the normal's label. This design choice enforces tasks to share network parameters even on the decoder side. Surprisingly, such simple design already offers us very competitive results.

During inference, only a small part of the embedding network performs computation. This allows our CompositeTasking network to use an efficient single encoder - single decoder architecture for all tasks. Furthermore, our training strategy enables users to request any task at any pixel. In fact, we also propose to learn the Task Palette, in case it is missing. The inferred palette follows some hand-crafted rules for task requests. It is then fed back to our network to execute the spatially distributed, rule-based tasks.

Learning pixel-specific tasking has several benefits, which may be obvious when a parallel to the image segmentation is drawn. In this work, we demonstrate the benefits in regard to a couple of chosen applications, namely, learning the Task Palette, task editing, and rule transfer. In the following, we summarize key contributions of our work.

- We introduce the new problem of CompositeTasking which we demonstrate to be useful for images.
- A novel method for CompositeTasking is also proposed. It is significantly superior in terms of computational efficiency, and competitive in terms of performance for image understanding tasks.
- Applications of the proposed CompositeTasking network, namely on predicting with an estimated Task Palette, task editing, and rule transfer, are also demonstrated in this paper.

2. Related Work

Multi-task learning (MTL). MTL is concerned with learning multiple tasks simultaneously, while exerting shared influence on model parameters. The potential benefits are manifold, and include speed-up of training or inference, higher accuracy, better representations, as well as higher parameter or sample efficiency. A comprehensive survey on architectures, optimization and other aspects of MTL can be found in [12]. On one hand, many MTL methods in the literature perform multiple tasks by a single forward pass, using shared trunk [8, 32, 54, 31, 14], cross talk [36], or prediction distillation [59, 65, 66, 55] architectures. On the other hand, the following methods perform one task at a time by conditioning a shared encoder, using feature masking [51], task-specific projections [67], attention mechanisms [34] or parametrized convolutions [22], while using one decoder head for each task. With CompositeTasking, we are bridging the gap between the two paradigms, by performing multiple tasks on the input image within one forward pass, by performing one task at a time – for each pixel. In stark contrast to conditioning a shared encoder, as done in [34, 22, 46, 5, 67, 51], we instead learn an unconditioned encoder, together with pixel-wise conditioning of a single unified decoder to output the task composition.

Conditional normalization (CN). Conditional normalization is the workhorse of many methods solving diverse problems in multi-domain learning [46, 47], image generation [23, 9, 43], image editing [42], style transfer [19], and super-resolution [56]. The operating principle is as simple as applying condition-dependent affine transformations on the normalized batch [21], local response [26], instance [52], layer [4], or feature group [58], allowing features to occupy different regions in the space, depending on the triggered condition. While many of the aforementioned tasks however only require a conditioning on image level – encoding for instance domain, class, or style – we are in need to perform pixel-wise varying conditioning. Inspired by the success of dense conditioning for semantic

image synthesis, we realise our pixel-wise task conditioning via spatially-adaptive normalization [56, 43]. To our knowledge, we are presenting the first method that learns multiple tasks with a single conditional unified head for all tasks, and is even so capable of performing multiple tasks – for different regions – in one forward pass.

Learning from partial labels. Crowdsourcing platforms such as Amazon Mechanical Turk or reCAPTCHA made image annotation affordable, and have brought valuable contributions for the computer vision community [49, 30, 33, 64]. In order to make best use of all annotators, an efficient approach is required for large-scale labelling, which may come at the price of only obtaining partial labels for each image. This trade-off is still favorable, since partially annotating more images typically outperforms dense labelling of fewer images, due to the increased variety of images seen during training [15]. Although the partial label problem has been addressed for diverse tasks such as segmentation [60, 3], depth densification [45], or multi-label classification [15, 20], it has only been tackled from the perspective of a single task at a time. In contrast, with our approach one can handle partial annotations of multiple tasks in the same image. Offering the ability to focus the learning of interesting tasks in interesting regions can significantly boost sample efficiency during training.

3. CompositeTasking Network

In this paragraph, we introduce the formal notations. The input image is denoted as $\mathcal{I} \in \mathbb{R}^{3\times H\times W}$, where H is height and W is the width of an image. The image is represented using 3 color channels. Next, we introduce the Task Palette, which is denoted as $\mathcal{T} \in [1, ..., K]^{H\times W}$. It is of the same spatial dimension as the input image $H \times W$, and it takes one of K discrete values, where K is the number of considered prediction tasks. The Task Palette specifies which task to predict at which pixel location. The model takes the image $\mathcal{I}$ and the Task Palette $\mathcal{T}$ as inputs and produces $\mathcal{O} = \mathsf{M}(\mathcal{I}, \mathcal{T})$, where $\mathcal{O} \in \mathbb{R}^{3\times H\times W}$. The output has the same spatial dimension as the input image $H \times W$ and also has 3 output channels. To construct the output $\mathcal{O}$, the model predicts task t_{yx} at output location o_{yx}. The output $\mathcal{O}$ is called the Composite Task, since it is a spatial composition of considered tasks $1, .., K$. Pixel-wise, every task is represented as a 3D vector $o_{yx} \in \mathbb{R}^3$. An overview of our architecture is presented in Figure 2. A detailed network diagram can be found in the supplementary materials.

3.1. Network Overview

The proposed model is divided into two parts: the encoder and the decoder network. This is inspired by the U-net [48] architecture. The encoder only processes the input image $\mathcal{I}$. The decoder takes the features processed by the encoder, along with the Task Palette $\mathcal{T}$, to produce the output $\mathcal{O}$. The encoder's job is to learn to produce a very rich feature representation that is sufficient to predict all K tasks. This representation is expected to capture enough information to be translated into different spatial compositions of tasks. The decoder's job is to take that feature representation, as well as a specific Task Palette $\mathcal{T}$, and translate it into the output $\mathcal{O}$. For the sake of simplicity, we choose 3 channels for the output $\mathcal{O}$ to treat this problem as image-to-image translation. Choice for higher number channels can also be made, if needed. Such choice is made merely for the convenience, which is also empirically supported.

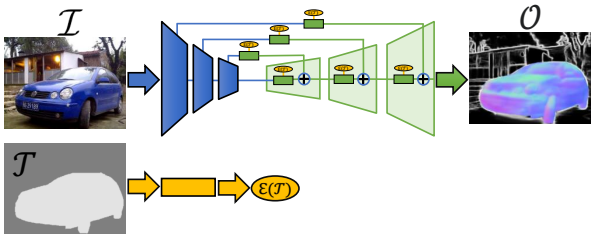

Figure 2: **Overview of the CompositeTasking network architecture.** We follow a U-NET [48] like design for Image-to-image translation. Blue blocks are modules from the encoder backbone. Green blocks of the decoder are depicted in Figure 4. The yellow block which produces the Task Palette embedding $\mathcal{E}(\mathcal{T})$ is depicted in Figure 3.

Having the same number of output channels for each task, as well as the ability to predict different tasks at different spatial locations, allows us to have a truly multi-tasking network. This means that the exact same network can predict any considered task at any considered location with the exact same architecture. We believe that most of the visual tasks can also be embedded within few channels. In fact, a similar practice is common in the domain of instance segmentation [41, 13, 7, 27, 17, 39, 40] and an another related work [1]. Our image format-based output chooses some arbitrary embedding for every task. We aim to map images to such embedding, conditioned upon the spatially distributed task requests. The task specific losses are then computed by mapping the predicted embedding to the task-appropriate label representations. This allows our network to produce the same output format, thereby making addition of new tasks very simple. More importantly, additional tasks require no additional modules, network heads, etc. The architecture of the CompositeTasking network, as well as its parameter count remain exactly the same.

3.2. Conditioning with the Task Palette

The spatially specific output conditioning is achieved by using a layer described in this section. This layer is inspired by [43], which was originally used to generates images with desired semantic structure. We however, perform BatchNorm normalization [21] only in the decoder, where task

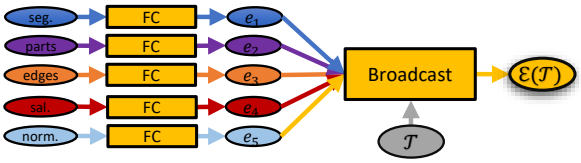

Figure 3: **Task representation block.** Each task is process independently to learn the task-specific embedding. We broadcast these embeddings according to the task request $\mathcal{T}$. The broadcast $\mathcal{E}$ is fed into the composition blocks from Figure 4.

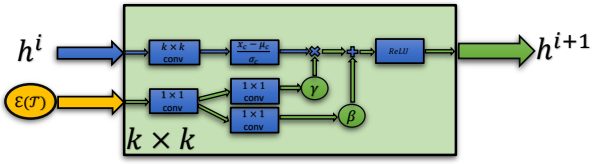

Figure 4: **Task composition block**. This block receives task embedding and previous layer's features as inputs, and performs task-conditioned transformation of the features.

specific affine transformation parameters are predicted differently for each pixel conditioned upon the Task Palette $\mathcal{T}$ at that spatial location t_{yx}. For the notational convenience, we follow [43]. Let $\mathbf{h}^i$ denote the activation of layer i in the decoder, for a batch of N samples. Let H^i and W^i be the height and width of the activation map and let C^i be the number of channels in layer i. Then, the task specific conditioning is achieved by using a layer which computes,

$$h^{i+1}_{ncyx} = \gamma^i_{cyx}(t_{yx}) \frac{h^i_{ncyx} - \mu^i_c}{\sigma^i_c} + \beta^i_{cyx}(t_{yx}), \qquad (1)$$

where h^{i+1}_{ncyx} is the output of our task-specific conditioning layer, and μ^i_c and σ^i_c are the mean and standard deviation of the activations in channel c:

$$\mu^i_c = \frac{1}{NH^iW^i} \sum_{n,y,x} h^i_{ncyx}, \qquad (2)$$

$$\sigma^i_c = \sqrt{\frac{1}{NH^iW^i} \sum_{n,y,x} \left((h^i_{ncyx})^2 - (\mu^i_c)^2\right)}. \qquad (3)$$

The affine parameters $\gamma^i_{cyx}(t_{yx})$ and $\beta^i_{cyx}(t_{yx})$ condition the normalized activation h^i_{ncyx} based on the requested tasks t_{yx}. Unlike [43], which allows surrounding semantics dependent γ^i_{cyx} and β^i_{cyx} by using 3×3 convolutions on conditioned semantics, we keep operations for each task request independent. In other words, during conditioning, [43] aims to fit pixels meaningfully in the surrounding, whereas we are interested to make each request independent. This choice is motivated by the potential of flexible applications of our method, some of which are demonstrated later in this paper. In this process, we first transform the Task Palette to an embedding $\mathcal{E} = f(\mathcal{T})$, where $\mathcal{E} \in \mathbb{R}^{H \times W \times N_w}$. The parameters $\gamma^i_{cyx}(e_{yx}(t_{yx}))$ and $\beta^i_{cyx}(e_{yx}(t_{yx}))$ are then obtained using the embedding $\mathcal{E}$. Here, we present further details of our method using two blocks: (a) task representation; and (b) task composition.

Task representation block. In order to embed the Task Palette $\mathcal{T}$ into $\mathcal{E} = f(\mathcal{T})$, we first learn the task specific embeddings $\{e_1, e_2, \ldots, e_K\}$ for each task. This is done by embedding all distinct values of the Task Palette $e_k = f(z_k)$, where z_k is the unique task code. The Palette's embedding $\mathcal{E}$ is then obtained by broadcasting task specific embeddings according to the task requests $\mathcal{T}$. All of the Task Composition Blocks use the same embedding $\mathcal{E}$. In Figure 3, we can see that each task is processed independently through a fully connected Neural Network, before broadcasting into $\mathcal{E}$.

Task composition block. A graphical representation of this block depicted in Figure 4. The task composition block receives the task embedding $\mathcal{E}$ and the features from the previous layer of the network. The conditioning operation of (1) takes place within this block as follows: (i) features are processed using a standard convolution layer; (ii) embedding $\mathcal{E}$ is processed independently for each task using two layers of 1×1 convolutions to obtain $\gamma^i_{cyx}(e_{yx}(t_{yx}))$ and $\beta^i_{cyx}(e_{yx}(t_{yx}))$; (iii) the operation of (1) is then performed followed by an activation function. Output of this block is the task-conditioned transformed features. As shown in Figure 2, we use the task composition blocks only in the decoder and skip connections of our network.

3.3. Computing the Loss and Training

Every task k has its own loss function of interest $\mathcal{L}_k$. The total loss is given by:

$$\mathcal{L} = \sum_k \lambda_k \mathcal{L}_k(\mathcal{O}, \mathcal{Y}_k, \mathcal{T}), \qquad (4)$$

where $\mathcal{Y}_k$ are the labels for task k. Usually the losses are computed per pixel location as $\mathcal{L}_k(o_{yx}, y_{yx})$, but that doesn't necessarily have to be the case. The hyperparameter λ_k helps to define the balance/trade-off of the predictive performance of all K tasks under consideration. The CompositeTasking network uses a standard training procedure. The images from the training set $\mathcal{I}$ are continuously passed to the network as inputs, along with the desired Task Palettes $\mathcal{T}$ to predict the output $\mathcal{O} = \mathsf{M}(\mathcal{I}, \mathcal{T})$. The predicted output $\mathcal{O}$ is given to the loss function (4), along with the corresponding labels $\mathcal{Y}_1, ..., \mathcal{Y}_K$ and Task Palette $\mathcal{T}$. The final loss is minimized using standard optimization algorithms [25].

4. Tasks and Rules

We consider the following dense prediction tasks.

Semantic segmentation. This is the task of predicting which of the defined semantic classes the pixel belongs to.

Human body parts. Similar to semantic segmentation, this is the task of predicting which of the defined human body parts the pixel belongs to.

Surface normals. This is the task of predicting the 3D orientation of the surface contained in the pixel.

Semantic Edges. This is the task of predicting edges between different objects on the input image.

Saliency. This is the task of predicting which locations in the image are most conspicuous for human observers.

Since the network output is constrained to 3 channels, we need to embed the tasks accordingly. Surface normals are 3D vectors by nature, and fit in the output shape. In the case of edges and saliency, the output corresponds to a scalar probability of the positive class. One way to embed them is to predict them at all 3 channels and calculate the mean at test-time. For the task of semantic segmentation and human parts, the pixel-wise outputs are usually represented as length C vectors representing the probability of each class. To this end, we transform the pixel-wise 3D output $\mathbf{o}$. First, we define 3D class anchors $\mathbf{a}_i$ to each class i, uniformly spread out in space (more details in the supplementary materials). Then, we compute a score vector $\mathbf{l} \in \mathbb{R}^C$ based on the distance to the class centers,

$$l_i = \frac{1}{\|\mathbf{o} - \mathbf{a}_i\| + \epsilon}, \tag{5}$$

where ϵ is a small constant. Hence, $\mathbf{l}$ has the highest value at the index of the closest class center. Applying a softmax operation $\hat{o}_i = \frac{e^{l_i}}{\sum_{j=1}^{C} e^{l_j}}$ transforms $\mathbf{l}$ to a probability measure that can be used with the common loss functions. One can argue that if we look at $\hat{o}$ as the predicted class-probabilities, it is biased by the arbitrary definition of class anchors. If the class i has the highest predicted probability $i = \text{argmax}_j \hat{o}_j$, only one of the classes $\{j : a_j \in \mathcal{N}(\hat{o}_i)\}$ close to $\hat{o}_j$ can have the second highest probability. Therefore we can not analyze class similarity by looking at these predicted probabilities. However, since our fist priority for the segmentation tasks is predicting the correct class, this offers a simple and effective solution. In case one is interested in making more detailed conclusions from the prediction, alternative approaches can be taken.

The rules for constructing Task Palettes $\mathcal{T}$ for our experiments are as follows.

Single task rule $\mathcal{S}$: The Task Palette $\mathcal{P}$ has the same value k at every location $\forall x, y : t_{yx} = k$. Task k can be changed every time the Task Palette is requested from this rule.

Random mosaic rule $\mathcal{R}_{1r}$: The image is spatially divided into four rectangles by intersecting a vertical and horizontal line through a randomly chosen point $\mathbf{c} = (c_x, c_y)$. Each region gets a task assigned to it randomly. The assigned tasks, as well as the point $\mathbf{c}$, can be changed every time the Task Palette is requested from this rule.

Semantic rules $\mathcal{R}_2$ **and** $\mathcal{R}_3$: These rules assign the tasks with respect to the image semantics.

Random rule $\mathcal{R}_{rnd}$: This rule assigns a randomly chosen task to each pixel independently.

More details on these rules can be found in the supplementary materials.

Our rationale for choosing these rules is based on our desire to analyse the behaviour of our method. Rule $\mathcal{S}$ is used in the field for solving specific problems. Rule $\mathcal{R}_{1r}$ is one way of seeing what happens if you train and test the network by mixing tasks in the output randomly, without any specific rule or structure behind it. Rules $\mathcal{R}_2$ and $\mathcal{R}_3$ represents rules with some semantic meaning behind it. Finally, rule $\mathcal{R}_{rnd}$ is designed to test the proposed method's limits.

5. Implementation Details

5.1. Data Set Description

The experiments are conducted on the PASCAL-MT data set from [34]. While constructing the data set authors distilled labels for some of the tasks, while others were used from PASCAL [16] or PASCAL-Context [38]. The data set contains 4998 training and 5105 validation images. We predict the tasks mentioned in section 4. We evaluate the performance of semantic segmentation and human body parts with the mean intersection over union (mIoU). We evaluate the performance of saliency with computing the maximal mIoU over different thresholds. We compute the prediction of surface normals as the mean angular error (mErr). And finally, we evaluate the performance of edges with the optimal dataset F-measure (odsF) [35], using the implementation from [44]. These evaluation metrics are in concordance with recent multi-tasking work [34, 22].

5.2. Experimental Setup

In our experiments we use the following models:

CompositeTasking network (CTN). This is the network we proposed in section 3. The network uses a ResNet34 [18] encoder. The decoder is build using spatially varying conditioning blocks from Figure 4, and it is much smaller than the encoder, in terms of network parameters. The conditioning blocks use 1×1 regular convolutions in the skip connections and 3×3 elsewhere, which performed well empirically.

Single task networks baseline (STN). Here we have different networks for different tasks. Each network has the same architecture as the CompositeTasking network, but instead of the spatially varying conditioning, they use a regular BatchNorm. This way the network has the same capac-

ity. In the case of the single tasking rule $\mathcal{S}$ from section 4, each task will be supervised by a different network. With the other CompositeTasking rules $\mathcal{R}_{1r}$ and $\mathcal{R}_2$ it is going to do the same, which means that the network for a specific task will only be supervised on the pixels that correspond to that task in the Task Palette $\mathcal{T}$.

Multi-head network (MHN). Here we have a network with a shared encoder, and a different decoder for different tasks. This is a standard approach in multi-task learning [46, 5, 34, 22]. The encoder is the same as in the CompositeTasking network, while the decoders have the same architecture, but use regular BatchNorm instead of the task specific conditioning. Similarly as above, this network is going to be supervised by supervising each decoder only on the pixels from its corresponding task.

Since CompositeTasking is a new concept, we decided to evaluate the performance of our proposed CompositeTasking network (Figure 2) with standard baselines. The STN is a common pipeline for solving specific tasks in computer vision, while the MHN is a common pipeline for solving multiple tasks simultaneously in a multi-tasking fashion.

For more implementational details and hyper-parameter values look at the supplementary materials.

6. Experiments

6.1. General Behaviour

To compare the performance of a method m with baseline models from Section 5.2, we use the average per-task drop with respect to the single-tasking baseline b, $\Delta_m = \frac{1}{T}\sum_{i=1}^{T}(-1)^{l_i}\frac{M_{m,i}-M_{b,i}}{M_{b,i}}$, where $l_i = 1$ if a lower value is better for measure M_i of task i, and 0 otherwise [34].

We first evaluate on the single-task rule $\mathcal{S}$. From Table 1, we can see that the CompositeTasking network is on par with the baselines, even though it is trained with randomly cropped label regions of rule $\mathcal{R}_{1r}$, and tested on the single-task setting $\mathcal{S}$. This is very interesting, since it is substantially more compact in terms of memory and computational complexity, as can be seen in Figure 5. Also, this experiment tells us that a lot can be learned even if only arbitrary regions for labels are being presented during training, as is the case with rule $\mathcal{R}_{1r}$. More interestingly, the part of the label that is being presented during training is a random rectangle region without any semantic meaning behind the chosen region, and still the network performs competitively to the strongest multi-head baseline trained on complete labels. A few examples of the networks predictions are presented in Figures 6 and 7. More examples are presented in the supplementary materials. We can see that it poses no problem for the network to sharply switch from predicting one task to another with negligible boundary artifacts, while using the exact same architecture to predict different tasks. For reference, the results from Table 1 can be compared to SotA baselines in [34] (Tables 2 and 3) and [22] (Table 3).

Table 1: **Testing the models on the single task rule**. Our method achieves the same performance as the baselines with much more capacity, when trained on randomly cropped label regions $\mathcal{R}_{1r}$.

Training rule	Method	Edge↑	SemSeg↑	Parts↑	Normals↓	Sal↑	Δ_m%↓
$\mathcal{S}$	STN	69.50	63.69	58.76	15.58	69.38	0.0%
	MHN	68.10	60.77	54.21	16.44	67.21	-4.60 %
$\mathcal{R}_{1r}$	STN	68.30	59.82	49.88	16.07	69.94	-5.05%
	MHN	67.70	61.64	52.84	16.40	67.70	-4.71%
	CTN(Ours)	68.60	62.45	52.59	16.93	67.81	-4.93%

Table 2: **Testing the models on the semantic rule**. Our method achieves the same performance as the baselines with much more capacity, when trained on the semantic rule $\mathcal{R}_2$.

Training rule	Model	Edge↑	SemSeg↑	Parts↑	Normals↓	Sal↑	Δ_m%↓
$\mathcal{S}$	STN	65.80	79.32	56.21	14.75	73.23	0.0%
	MHN	64.60	73.94	50.65	15.68	71.01	-5.57%
$\mathcal{R}_{1r}$	STN	64.80	74.12	44.81	15.42	73.22	-6.58%
	MHN	65.20	75.97	49.07	15.62	71.74	-5.15%
	CTN(Ours)	62.30	76.73	48.87	16.45	71.40	-7.13%
$\mathcal{R}_2$	STN	63.90	83.91	59.63	17.13	70.04	-2.30%
	MHN	64.40	83.71	58.44	17.44	67.02	-3.87 %
	CTN(Ours)	69.20	84.70	59.74	18.12	67.95	-2.37%

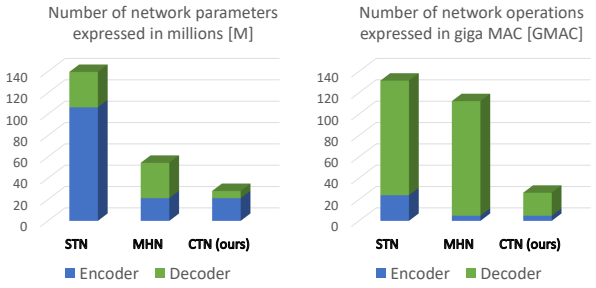

Figure 5: **Model Complexity.** Memory and computational requirements of the compared methods for predicting 5 tasks.

In Table 2, we see the performance evaluated using the semantic rule $\mathcal{R}_2$ where the tasks are being requested only at sparse, but meaningful and compact regions for each task. When we supervise by using rule $\mathcal{R}_2$, again we see that the CompositeTasking network performs on par with the baselines that have a lot more parameters. This shows that it is not necessary to waste so much resources when dealing with very sparse labels. The performance of our model is almost the same as the much more demanding baseline, that has a separate network for each task.

6.2. Learning What to Do Where

Up until now we have considered cases when it is known what tasks to predict where (the Task Palette $\mathcal{T}$ is known). This is definitely interesting in some use-cases, like for example Augmented Reality applications where a user can specifically request what he wishes the algorithm to do. It is even more interesting for the Task Palette to be predicted by the network itself, given the input image. One such example is when we have a semantic rule like $\mathcal{R}_2$, where for every image we can supervise what needs to be predicted where.

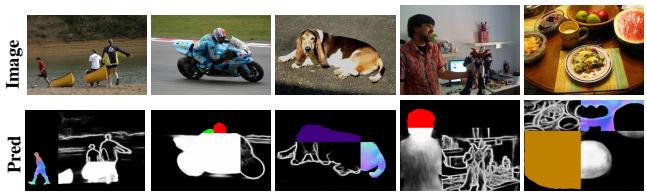

Figure 6: **Random mosaic compositions.** CompositeTasking network predictions on requests with the $\mathcal{R}_{1r}$ rule. The network shows the ability to sharply switch to a different tasks at neighbour pixels.

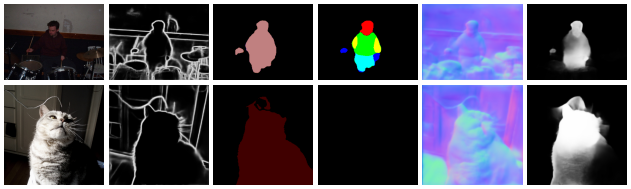

Figure 7: **Single task predictions.** Even though our model is made for CompositeTasking, it can also make predictions on requests with the $\mathcal{S}$ rule.

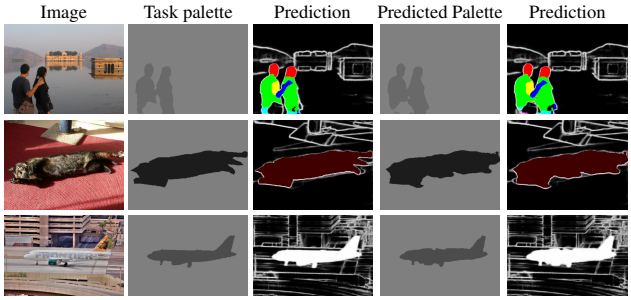

Figure 8: **Learning what to do where.** CompositeTasking network predictions with the learned Task Palette. A separate network is learned to predict the Task Palette with 75.04% mIoU. More examples can be found in the supplementary materials.

We trained a network to predict the Task Palette from the input image (75.04% mIoU), during supervision with $\mathcal{R}_2$. This can be used for automatic data labelling when we are interested in obtaining labels for multiple different tasks, but only in sparse regions of interest.

6.3. Task Palette Editing

The world is striving towards automated processes, but things are not perfect just yet. Data labelling is a very unpleasant and time-consuming job if someone has to make dense annotations. Very often we are interested in labels of different tasks at different spatial locations. For example, we want to predict surface normals of cars so that we can perform realistic re-rendering, at the same time semantic segmentation of the surrounding objects and edges everywhere else. That is a very clear rule that can be learned by

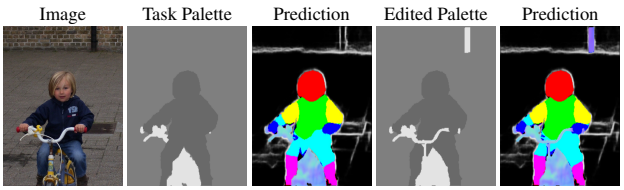

Figure 9: **Task Palette editing.** The original Task Palette, extracted from the label, did not look satisfactory. A correction was made manually. A prediction of the CompositeTasking network is shown before and after correction.

the setup proposed in 6.2. This will not be perfect however, and from time to time mistakes will be made. In such scenarios there may be a human in the loop, making sure that all mistakes along the execution pipeline are corrected. Our framework offers to take the predicted Task Palette, where most of it is correct, and edit the mistakes in certain regions. In that setup, most of the work is done automatically, and the human in the loop puts her focus mostly on regions of high interest to the use-case. One such visual example is presented in Figure 9, while more can be found in the supplementary materials. This highlights further the flexibility of our proposed method.

6.4. Breaking the Rule

"The only constant in life is change", as Heraclitus once wisely said. In fact, also prediction tasks are constantly evolving and are subject to change according to use case, context or available resources. One could think of an existing task rule, say $\mathcal{R}_2$, needs to be adapted to cater for a changing use case that demands a new rule $\mathcal{R}_3$. It can be similar in one way, but different in another compared to $\mathcal{R}_2$. For instance, the same tasks are used from $\mathcal{R}_2$, but now $\mathcal{R}_3$ requires different tasks to be performed in different regions.

One practical example of this is predicting surface normals. Often, the accurate normal labels are obtained by having accurate 3D models. The 3D models however may cover only a part of the scene, therefore of the image. This builds a rule of having normals only for the objects with 3D models. In fact, similar datasets exist. For example, datasets with 3D models of household objects like chairs and tables [28], and of the human body [6]. Using the model trained on such datasets, one may be interested to predict normals beyond the reason of 3D models. Here we show that our CompositeTasking network can indeed be trained by breaking the rule.

We break the old rule by simply requesting to execute the task of the new rule. This is then followed by fine-tuning our network on the new rule, if necessary. As shown in Table 3, our model trained on $\mathcal{R}_2$ is already doing good on a newly introduced rule $\mathcal{R}_3$, without any fine-tuning. The rule $\mathcal{R}_3$ is somewhat similar to $\mathcal{R}_2$, and the model shows the

Table 3: **Breaking the rule**. Our method can make successful predictions when changing from $\mathcal{R}_2$ to a somewhat similar rule $\mathcal{R}_3$ (described in the supplementary materials). With fine-tuning on the new rule, the predictions get even better.

Testing rule	Training rule	Edge↑	Parts↑	Normals↓	Sal↑	Δ_m%↓
$\mathcal{R}_3$	$\mathcal{R}_3$	70.20	61.19	18.34	75.35	0.0%
	$\mathcal{R}_2$	69.70	59.41	20.11	65.21	-6.68%
	Fine-tuned $\mathcal{R}_2 \rightarrow \mathcal{R}_3$	69.70	60.91	18.68	75.00	-0.87%
$\mathcal{R}_2$	$\mathcal{R}_2$	69.20	59.74	18.12	67.95	0.0%
	Fine-tuned $\mathcal{R}_2 \rightarrow \mathcal{R}_3$	69.40	60.84	17.95	68.31	+0.90%

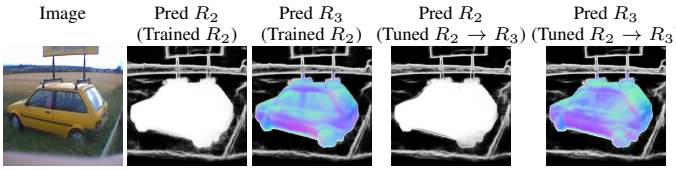

Figure 10: **Breaking the rule.** Predictions of the CompositeTasking network are show before and after fine-tuning from the old to the new semantic rule. Because the rules are similar in a way, the model can extrapolate even before fine-tuning.

Table 4: **Evaluating on randomly chosen tasks at each pixel location independently**. Notice the performance difference between training on $\mathcal{R}_{1r}$ vs. $\mathcal{R}_{rnd}$, and testing on $\mathcal{R}_{rnd}$. The edge evaluation is omitted due to the unsuitable evaluation protocol.

Trained on rule	Evaluated on rule	SemSeg ↑	Parts↑	Normals↓	Sal↑
$\mathcal{R}_{1r}$	$\mathcal{S}$	62.45	52.59	16.93	67.81
$\mathcal{R}_{1r}$	$\mathcal{R}_{rnd}$	35.89	21.11	64.36	66.71
$\mathcal{R}_{rnd}$	$\mathcal{R}_{rnd}$	59.58	52.28	17.16	67.60
$\mathcal{R}_{rnd}$	$\mathcal{S}$	52.26	51.88	22.65	65.34

ability to extrapolate on their differences. After fine-tuning it on $\mathcal{R}_3$, the performance improves even more, as expected. One example of this is presented in Figure 10, while more can be found in the supplementary materials. Interestingly, in Table 3 we observe that the old rule is even improved when training on the new similar rule.

6.5. Random Compositions

Finally, we are interested to see what happens if our model is evaluated on tasks chosen independently for each pixel at random, denoted as $\mathcal{R}_{rnd}$. Table 4 indicates that our model trained on the mosaic rule $\mathcal{R}_{1r}$ does not perform very well on the random rule $\mathcal{R}_{rnd}$. We conjecture this is because the rule $\mathcal{R}_{1r}$ assigns tasks only to large connected regions during training, and no incentive is given to learn the ability to switch tasks with high spatial frequency. Training on the random rule $\mathcal{R}_{rnd}$, consequently improves the performance on $\mathcal{R}_{rnd}$ significantly (Table 4). A visualization of this results is presented in Figure 11. Interestingly, although only using a single output, we can clearly observe a meaningful execution of different tasks all over the image.

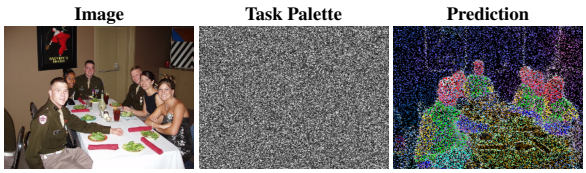

Figure 11: **Random tasks chosen at each pixel independently.** Predictions includes all five tasks of rule $\mathcal{R}_{rnd}$.

7. Discussion

We feel that this is only the tip of the iceberg. In [37], for example, it is crucial to fuse different tasks like face detection, pose estimation, scene understanding and depth estimation to obtain state-of-the-art performance for the high-level task of emotion recognition. Many state-of-the-art pipelines for high-level tasks share that approach [1] and more often than not, different predicted tasks feel important only at different spatial regions of the input image. A great potential is seen for CompositeTasking here. This compactness in terms of memory and computation efficiency can sometimes determine whether some solution to the problem can be practically implemented or not. Also, wasting resources when there is no need for it is never welcome.

While supervising the high-level predictions, one can also attempt to learn the rule of what is beneficial to predict where, even if such a rule is not known a priori. This can bring a new level of understanding how the very complex Deep Learning models make decisions on high-level tasks, by observing the requests that the network is making during inference.

8. Conclusion

In this work, we introduced the concept of CompositeTasking as the fusion of multiple, spatially distributed tasks, motivated by the frequent availability of only sparse labels across tasks, and the desire for a compact multi-tasking network. To this end, we studied a novel task conditioning model – a single encoder-decoder network that performs multiple, spatially varying tasks at once. We showed that CompositeTasking offers efficient multi-task learning from only sparse supervision, with performance competitive to dense supervision and a multi-headed multi-tasking design. Moreover, we demonstrated the unique flexibility by our approach with regards to interactive task editing, and rules transformations.

[1] Andrej Karpathy, senior director of AI at Tesla, recently said that they use a multi-tasking system with 48 shared backbones and 1000 different output task heads in their self-driving Autopilot high-level task

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6430–6439, 2019.

[2] Somak Aditya, Yezhou Yang, and Chitta Baral. Integrating knowledge and reasoning in image understanding. *arXiv preprint arXiv:1906.09954*, 2019.

[3] Iñigo Alonso, Ana B. Cambra, A. Muñoz, T. Treibitz, and A. C. Murillo. Coral-segmentation: Training dense labeling models with sparse ground truth. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2874–2882, 2017.

[4] Jimmy Ba, J. Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[5] Hakan Bilen and A. Vedaldi. Universal representations: The missing link between faces, text, planktons, and cat breeds. *ArXiv*, abs/1701.07275, 2017.

[6] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Piscataway, NJ, USA, June 2014. IEEE.

[7] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *CoRR*, abs/1708.02551, 2017.

[8] Felix J. S. Bragman, Ryutaro Tanno, Sébastien Ourselin, D. Alexander, and M. Cardoso. Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1385–1394, 2019.

[9] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *ArXiv*, abs/1809.11096, 2019.

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

[11] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12538–12547, 2019.

[12] M. Crawshaw. Multi-task learning with deep neural networks: A survey. *ArXiv*, abs/2009.09796, 2020.

[13] B. De Brabandere, D. Neven, and L. Van Gool. Semantic instance segmentation for autonomous driving. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 478–480, 2017.

[14] C. Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2070–2079, 2017.

[15] T. Durand, Nazanin Mehrasa, and G. Mori. Learning a deep convnet for multi-label classification with partial labels. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 647–657, 2019.

[16] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[17] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P. Murphy. Semantic instance segmentation via deep metric learning. *CoRR*, abs/1703.10277, 2017.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[19] X. Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1510–1519, 2017.

[20] D. Huynh and E. Elhamifar. Interactive multi-label cnn learning with partial labels. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9420–9429, 2020.

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

[22] M. Kanakis, David Bruggemann, Suman Saha, S. Georgoulis, Anton Obukhov, and L. Gool. Reparameterizing convolutions for incremental multi-task learning without task interference. *ArXiv*, abs/2007.12540, 2020.

[23] Tero Karras, S. Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2019.

[24] Seung Wook Kim, Makarand Tapaswi, and Sanja Fidler. Visual reasoning by progressive module networks. In *International Conference on Learning Representations*, 2018.

[25] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[26] A. Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60:84–90, 2017.

[27] Xiaodan Liang, Liang Lin, Yunchao Wei, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. Proposal-free network for instance-level object segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2978–2991, 2018.

[28] Joseph J. Lim, Hamed Pirsiavash, and Antonio Torralba. Parsing IKEA Objects: Fine Pose Estimation. *ICCV*, 2013.

[29] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[30] Tsung-Yi Lin, M. Maire, Serge J. Belongie, James Hays, P. Perona, D. Ramanan, Piotr Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. *ArXiv*, abs/1405.0312, 2014.

[31] Shikun Liu, Edward Johns, and A. Davison. End-to-end multi-task learning with attention. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1871–1880, 2019.

[32] Y. Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, T. Javidi, and R. Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1131–1140, 2017.

[33] S. Manen, Michael Gygli, Dengxin Dai, and L. Gool. Pathtrack: Fast trajectory annotation with path supervision. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 290–299, 2017.

[34] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *CVPR*, 2019.

[35] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.

[36] I. Misra, Abhinav Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3994–4003, 2016.

[37] Trisha Mittal, Pooja Guhan, Uttaran Bhattacharya, Rohan Chandra, Aniket Bera, and Dinesh Manocha. Emoticon: Context-aware multimodal emotion recognition using frege's principle, 2020.

[38] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[39] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Fast scene understanding for autonomous driving. *CoRR*, abs/1708.02550, 2017.

[40] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8837–8845. Computer Vision Foundation / IEEE, 2019.

[41] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2277–2287, 2017.

[42] Evangelos Ntavelis, Andrés Romero, Iason Kastanis, Luc Van Gool, and Radu Timofte. Sesame: Semantic editing of scenes by adding, manipulating or erasing objects, 2020.

[43] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[44] Jordi Pont-Tuset and Ferran Marques. Measures and meta-measures for the supervised evaluation of image segmentation. In *Computer Vision and Pattern Recognition*, 2013.

[45] Jiaxiong Qiu, Zhaopeng Cui, Y. Zhang, Xingdi Zhang, Shuaicheng Liu, B. Zeng, and M. Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3308–3317, 2019.

[46] 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.

[47] Sylvestre-Alvise Rebuffi, Hakan Bilen, and A. Vedaldi. Efficient parametrization of multi-domain deep neural networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8119–8127, 2018.

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICAI*, 2015.

[49] Olga Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Zhiheng Huang, A. Karpathy, A. Khosla, Michael S. Bernstein, A. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015.

[50] Ganesh Sistu, Isabelle Leang, Sumanth Chennupati, Senthil Yogamani, Ciarán Hughes, Stefan Milz, and Samir Rawashdeh. Neurall: Towards a unified visual perception model for automated driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 796–803. IEEE, 2019.

[51] Gjorgji Strezoski, Nanne van Noord, and M. Worring. Many task learning with task routing. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1375–1384, 2019.

[52] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *ArXiv*, abs/1607.08022, 2016.

[53] Sjoerd van Steenkiste, Francesco Locatello, Jürgen Schmidhuber, and Olivier Bachem. Are disentangled representations helpful for abstract visual reasoning? In *Advances in Neural Information Processing Systems*, pages 14245–14258, 2019.

[54] Simon Vandenhende, Bert De Brabandere, and L. Gool. Branched multi-task networks: Deciding what layers to share. *ArXiv*, abs/1904.02920, 2019.

[55] Simon Vandenhende, S. Georgoulis, and L. Gool. Mti-net: Multi-scale task interaction networks for multi-task learning. In *ECCV*, 2020.

[56] Xintao Wang, K. Yu, C. Dong, and Chen Change Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 606–615, 2018.

[57] Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding*, 163:21–40, 2017.

[58] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

[59] D. Xu, Wanli Ouyang, X. Wang, and N. Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. *2018*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 675–684, 2018.

[60] Zhenqi Xu, Shan Li, and Weihong Deng. Learning temporal features using lstm-cnn architecture for face anti-spoofing. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 141–145. IEEE, 2015.

[61] Xin Ye and Yezhou Yang. From seeing to moving: A survey on learning for visual indoor navigation (vin). *arXiv preprint arXiv:2002.11310*, 2020.

[62] Anthony M Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1):1–7, 2019.

[63] Richard Zhang. Making convolutional networks shift-invariant again, 2019.

[64] Richard Zhang, Phillip Isola, Alexei A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

[65] Z. Zhang, Zhen Cui, Chunyan Xu, Zequn Jie, Xiang Li, and Jian Yang. Joint task-recursive learning for semantic segmentation and depth estimation. In *ECCV*, 2018.

[66] Z. Zhang, Zhen Cui, Chunyan Xu, Yan Yan, N. Sebe, and J. Yang. Pattern-affinitive propagation across depth, surface normal and semantic segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4101–4110, 2019.

[67] Xiangyun Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. A modulation module for multi-task learning with applications in image retrieval. In *ECCV*, 2018.