International Journal of Computer Vision (2021) 129:3136–3153
https://doi.org/10.1007/s11263-021-01498-0

Semantic Bottlenecks: Quantifying and Improving Inspectability of Deep Representations

Max Losch[1] · **Mario Fritz**[2] · **Bernt Schiele**[1]

Received: 24 January 2021 / Accepted: 1 July 2021 / Published online: 14 September 2021

Abstract
Today's deep learning systems deliver high performance based on end-to-end training but are notoriously hard to inspect. We argue that there are at least two reasons making inspectability challenging: (i) representations are distributed across hundreds of channels and (ii) a unifying metric quantifying inspectability is lacking. In this paper, we address both issues by proposing Semantic Bottlenecks (SB), which can be integrated into pretrained networks, to align channel outputs with individual visual concepts and introduce the model agnostic Area Under inspectability Curve (AUiC) metric to measure the alignment. We present a case study on semantic segmentation to demonstrate that SBs improve the AUiC up to six-fold over regular network outputs. We explore two types of SB-layers in this work. First, concept-supervised SB-layers (SSB), which offer inspectability w.r.t. predefined concepts that the model is demanded to rely on. And second, unsupervised SBs (USB), which offer equally strong AUiC improvements by restricting distributedness of representations across channels. Importantly, for both SB types, we can recover state of the art segmentation performance across two different models despite a drastic dimensionality reduction from 1000s of non aligned channels to 10s of semantics-aligned channels that all downstream results are based on.

Keywords Explainable AI · Inspectability · Interpretability · Semantic segmentation · Representation learning

1 Introduction

While end-to-end training is key to top performance of deep learning, learned intermediate representations with typical training methods remain opaque to humans. Furthermore, assessing inspectability has remained a fairly elusive concept since its framing has mostly been qualitative (e.g. saliency maps). Given the increasing interest in using deep learning in real world applications, inspectability and a quantification of such is critically missing.

Goals for Inspectability To address this, prior work on inspectability has proposed to improve the spatial coherency of activation maps Zhang et al. (2018) or to cluster representations to acquire outputs of low dimensionality either with supervision (Bucher et al. 2018) or without (Chen et al. 2019; Yeh et al. 2019). In contrast, we demand information in each channel to be represented by a single semantic concept. This is derived from a simple observation: distributed representations do not lend themselves to trivial interpretation (see bottom of Fig. 1). In order to reduce distributedness and improve inspectability we propose to adapt deep networks via three criteria. (i) Reduce the number of channels to a minimum, (ii) associate them with semantic concepts, and, at the same time, (iii) aim to lose as little overall performance as possible. While goal (i) is not necessary to reduce distributedness, it substantially reduces the amount of information that needs to be interpreted by a human observer. In our view such semantics based inspectability can be seen as a way towards achieving true interpretability of deep networks.

Semantic Bottlenecks We propose to implement these three goals via Semantic Bottlenecks (SBs). SBs are single (convolutional) layers that are easy to integrate into existing, pretrained, networks which map the highly distributed outputs to a semantically aligned space of reduced dimensionality. To achieve this alignment, SBs can be constructed either supervised or unsupervised by regularizing the dis-

Communicated by Zeynep Akata.

✉ Max Losch
mlosch@mpi-inf.mpg.de

[1] Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

[2] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

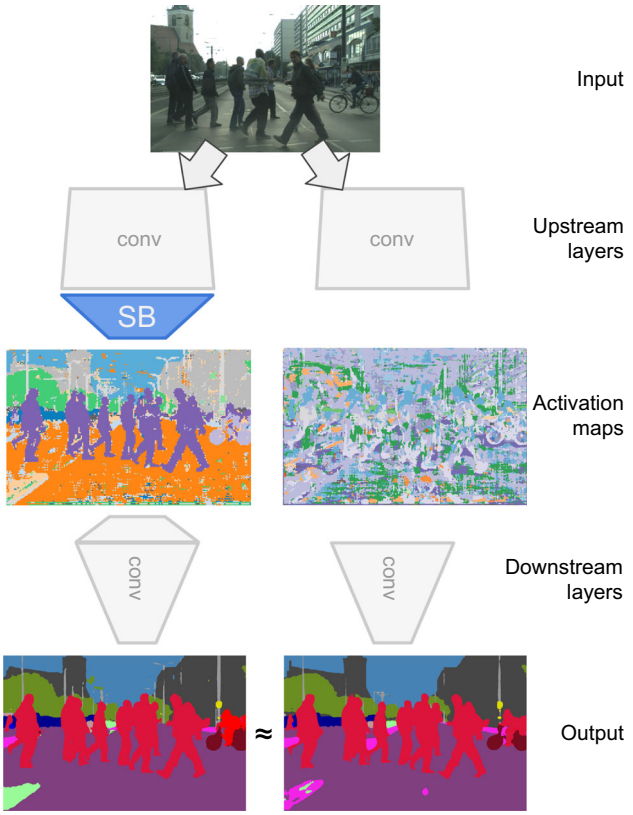


Fig. 1 Semantic Bottleneck (SB) layers can be integrated into any model and act as inspectable basis for subsequent layers. Their activation maps are coherent and well aligned with semantic concepts (left). The alternative, regular deep networks, have layers with highly distributed representations across hundreds of channels, making them hard to inspect (right)

tributedness of outputs and encourage the network to learn specialized features interpretable by humans. Our unsupervised SBs activate only a single channel for any given spatial location of its input map. While goal (ii), the semantic alignment, is not explicitly solvable in an unsupervised fashion, we find our regularization results in drastically improved channel inspectability. We show this qualitatively and more importantly: quantitatively with a metric comparing activation maps and concept annotations.

Our Contributions are three-fold. Firstly, we introduce two network layers we term Semantic Bottlenecks to improve alignment with semantic concepts as described in the last paragraph. The first—supervised SBs—improves alignment by aligning each channel with a single visual concept and the second—unsupervised SBs—align by restricting distributedness. Secondly, we show SBs can be integrated into two state-of-the-art models at early, intermediate or late layers without impairing performance, even when reducing the number of channels, e.g., from 4096 to 20. Finally, we introduce the novel Area Under inspectability Curve (AUiC) metric to quantify alignment between channel outputs and visual concepts for any model and show our SBs improve over baselines up to six-fold.

2 Related Work

As argued in prior work (Lipton 2018), interpretability can be largely approached in two ways. The first being post-hoc interpretation, for which an already trained and well performing model is dissected into its decisions a-posteriori to identify important (input) features via attribution. The second approach involves constructing inherently interpretable models, by aligning representations with semantic concepts—supervised or unsupervised.

Input Feature Attribution A large and diverse set of papers have been published on explaining deep network predictions by highlighting the most important input features. Early methods investigated the propagation of feature relevance from class logits back to input (Simonyan et al. 2014; Bach et al. 2015; Sundararajan et al. 2017). Others explored the use of input perturbation methods to determine the models sensitivity towards groups of pixels (Ribeiro et al. 2016; Lundberg and Lee 2017; Zintgraf et al. 2017; Petsiuk, et al. 2018; Fong et al. 2019). However, recent work has shown that many of these approaches are unfaithful to the model (Adebayo et al. 2018), decreasing trust in the method of input attribution. While some variants have been shown to be model faithful (Selvaraju et al. 2017; Srinivas and Fleuret 2019), input attributions entail two additional issues. First, they do not automatically increase human trust in the system as randomized control trials have indicated (Chu et al. 2020). Secondly, the accuracy of their explanations w.r.t. their predictions are difficult to measure. To overcome this, several benchmarks have been introduced to estimate the accuracy of the explanations (Samek et al. 2016; Hooker et al. 2019). These are based on removing or perturbing the most important input features, according to the respective attribution method. Assuming the input attribution is accurate, the altered training data should result in substantial drops in performance. The drawback: these tests are computationally demanding and need to be reevaluated for each individual model as attributions are model dependent.

In our work, we demand information within the model to be inspectable to improve interpretability of the whole process pipeline. Our semantic bottlenecks have a clear alignment with semantic concepts, presenting feature evidences by default.

Deep Feature Interpretability Input feature attribution does not offer direct insights into the inner working of deep nets. By means of analyzing individual representations, we can investigate what visual concepts a model is particularly sensitive to. The earliest work visualized features (Simonyan et al. 2014; Zeiler and Fergus 2014; Yosinski, et al. 2015;

Mahendran and Vedaldi 2015), later ones investigated statistical correlations between activations and input features (Bau et al. 2017; Kim et al. 2018; Chen et al. 2019). Given the distributedness of representations, it became evident that individual channels are difficult to associate with concise semantics (Fong et al. 2018). Hence, it was proposed to group activations before matching them (Li et al. 2018; Fong et al. 2018; Bau et al. 2019) or alternatively: combine semantics to match to individual channels (Mu and Andreas 2020). All of them serve a similar purpose: utilizing a well performing model and assigning semantics after training is finished.

In contrast, we design our semantic bottlenecks to have semantic meaning by design for which each channel is identifiable (not just partially Bau et al. 2017).

Supervised Semantic Alignment We are not the first to study semantic alignment by design in deep networks. Literature on supervised improvements has focused on embedding predefined semantic concepts based on auxiliary annotations (Li et al. 2015; Bucher et al. 2018; Marcos et al. 2019; Li et al. 2010). While Bucher et al. (2018) proposes a model based on natural language features, (Marcos et al. 2019) performs piece-wise training: pretraining on semantic concepts and training a secondary model on top for a related task. Most closely related to our supervised semantic bottlenecks are the just recently published concept bottleneck models (Koh et al. 2020). Similar to our approach, they insert a linear layer of drastically reduced dimensionality and supervise it to represent semantic concepts. While supporting our observation of increased inspectability, they test on image classification tasks only and do not extend their bottlenecks to unsupervised concept discovery.

We discuss in this paper supervised as well as unsupervised semantic bottlenecks, their application to various locations in a deep network and apply them to a safety critical task: street scene segmentation.

Unsupervised Semantic Alignment Similar to our USBs, (Al-Shedivat et al. 2020; Melis and Jaakkola 2018; Li et al. 2018; Yeh et al. 2019; Chen et al. 2019) embed an interpretable layer into the network using unsupervised metrics. Al-Shedivat et al. (2020), Melis and Jaakkola (2018), and Esser et al. (2020) base their method on a reconstruction loss that regularizes a latent code, that is used to reconstruct the input, to be more interpretable. Such approaches are based on the success of VAE (Kingma and Welling 2013) based frameworks and their recent success on learning visual concepts from simple datasets (Higgins et al. 2017; Greff et al. 2019; Burgess et al. 2019) but have the issue that reconstruction is more challenging than classification itself. An approach circumventing reconstructing inputs—invertible neural networks (Jacobsen et al. 2018)—has recently been utilized to create mappings to concept bottleneck layers without the necessity to retrain any parts of the original model (Esser et al. 2020). The main drawback: the unsupervised variant explored does not lend itself to simple inspectability. Channels need to be individually probed to reveal their otherwise hidden semantic similar to the latent code in VAEs.

Our proposed unsupervised semantic bottlenecks in comparison lend themselves to easy inspectability due to three reasons. First, they substantially reduce channel dimensionality, secondly they reduce distributedness across their channels and finally, they are easy to inspect since their outputs are one-hot encoded.

Quantification of Concept Alignment One critical issue obstructing effective research in the domain of interpretability is the lack of standardized metrics. Bau et al. addressed this issue with their seminal work "Network Dissection" (NetDissect)—a method counting number of channels assignable to single visual concepts (Bau et al. 2017). Here, edges of a bipartite graph, connecting channels and concepts, are weighted by measuring overlap between activation map and pixel annotations. This approach showed that a fraction of channels can be interpreted as individual concept detectors, yet also highlight that a large fraction of channels remain unidentifiable. Two additional variants enable concept association by combining channels (Fong et al. 2018; Bau et al. 2019). The most recent extension addresses the inverse problem: associating concept combinations to single channels (Mu and Andreas 2020). Our AUiC metric leverages the ideas of NetDissect and extends it to satisfy three criteria we deem important for measuring inspectability—which NetDissect does not satisfy.

In contrast to existing literature, we propose semantic bottlenecks which are easy to integrate in any architecture and offer inspectable outputs while retaining performance on a dense prediction task. Additionally, we introduce the model agnostic AUiC metric enabling benchmarking of inspectability.

3 Semantic Bottlenecks

To approach more inspectable intermediate representations we demand information in each channel to be represented by a single semantic concept. We propose two variants to achieve this goal: (i) supervise channels to represent unique concepts and (ii) restrict the distributedness of representations across channels to produce concept-specialized channels. We construct both variants as layers that can be integrated into pretrained models, mapping non-inspectable representations to an inspectable semantic space. We name these supervised and unsupervised Semantic Bottlenecks (SB).

Formal Definition Before we start introducing the supervised and unsupervised variants in detail, we first define the integration of SBs in an arbitrary network. For an architecture

with L-layers, we define its functional composition:

$$f_{\text{Net}}(x_0) = (f_L \circ f_{L-1} \circ ... \circ f_1)(x_0), \tag{1}$$

where x_0 defines the input to the first layer. For clarity let's also name the output of each layer:

$$x_i := (f_i \circ f_{i-1} \circ ... \circ f_1)(x_0), \tag{2}$$

where $0 < i \leq L$. We denote the output dimensionality of each layer as $x_i \in \mathbb{R}^{N_i}$ and design our SB such that it acts as bottleneck, reducing this dimensionality:

$$f_{\text{SB}} : \mathbb{R}^{N_i} \rightarrow \mathbb{R}^{N_{\text{SB}}}, \tag{3}$$

applied to a layer f_i, such that $N_{\text{SB}} < N_i$. To integrate an SB after layer f_i (where $0 < i < L$), we extend our network definition from above:

$$f_{\text{SBNet}}(x_0) = (f_L \circ ... \circ f_{i+1} \circ f_{\text{r}} \circ f_{\text{SB}} \circ f_i \circ ... \circ f_1)(x_0). \tag{4}$$

Note that we also integrated an additional layer f_{r} that ensures its output dimensionality matches the original layer: $f_{\text{r}} : \mathbb{R}^{N_{\text{SB}}} \rightarrow \mathbb{R}^{N_i}$. To emphasize which parameters of the network are subject to training, we finally define the following operator Θ:

$$\theta_i = \Theta(f_i), \tag{5}$$

which returns the set of all parameters θ for a given function f. How we setup f_{SB} and f_{r} and train them supervised or unsupervised is the subject of the following Sects. 3.1 and 3.2.

Case Study To show the utility of SBs, we choose street scene segmentation on the Cityscapes dataset (Cordts et al. 2016) since it is a difficult task that traditionally requires very large models and has a practical application that has direct benefits from inspectability: autonomous driving. Cityscapes consists of 19 different classes, 2 975 training images and 500 validation images, both densely labeled. We use PSPNet (Zhao et al. 2017) and the recent MS-OCRNet (Tao et al. 2020), both based on ResNet-101 (He et al. 2016), due to their strong performance on Cityscapes and because residual networks are abundantly used in computer vision tasks today. PSPNet uses a pyramid pooling module before classification and MS-OCRNet uses a hierarchical multi-scale object-contextual representation module (Yuan et al. 2020; Tao et al. 2020). Since both architectures are residual, we define their general functional composition w.r.t. their stages (also see Fig. 3):

$$f_{\text{ResNet}}(x_0) = \left(f_{\text{stage-}L} \circ f_{\text{stage-}(L-1)} \circ ... \circ f_{\text{stage-1}}\right)(x_0), \tag{6}$$

where a stage contains M residual blocks $\mathcal{B}$, all retaining the dimensionality:

$$f_{\text{stage-i}}(x_{i-1}) = (\mathcal{B}_M \circ \mathcal{B}_{M-1} \circ ... \circ \mathcal{B}_1)(x_{i-1}). \tag{7}$$

Each block implementing the residual function $\mathcal{B}(x) = \mathcal{F}(x) + x$, for a function $\mathcal{F}$.

3.1 Supervised Semantic Bottlenecks (SSBs)

SSBs (f_{SSB}) perform concept supervision on each SB channel using additional annotations. Ideally, we possess pixel-annotations for an exhaustive set of subordinate concepts like colors, textures and parts to decide which are required to recover performance on particular tasks. Yet, we show that an encouragingly small task-specific selection is sufficient to satisfy both desiderata of performance and inspectability.

Choosing Concepts for Cityscapes For our supervised SB-layer we choose concepts based on task relevancy for Cityscapes. *Broden+* (Xiao et al. 2018) is a recent collection of datasets which serves as a starting point for the concept annotations we require for the SSBs. It offers thousands of images for objects, parts, materials and textures for which the first three types come with pixel level annotations. Here, a pixel can have multi-label annotations. Based on the 377 part and material concepts available [351 parts sourced from ADE (Zhao et al. 2017) and Pascal-Part (Chen et al. 2014) and 26 materials sourced from OpenSurfaces (Bell et al. 2013)], we compile a subset of 70 Cityscapes-relevant concepts listed in Table 1.

Setup Let's specify the SSB setup formally for semantic segmentation. That is, our inputs and outputs are two dimensional. In our experiments, we compose the SSB to have two bottlenecks: one for materials, one for parts. In the general case, an SSB may have N bottlenecks, each containing channels of a specific concept category:

$$f_{\text{SSB}} := \Xi(f_{\text{cat-1}}, f_{\text{cat-2}}, ..., f_{\text{cat-}N}), \tag{8}$$

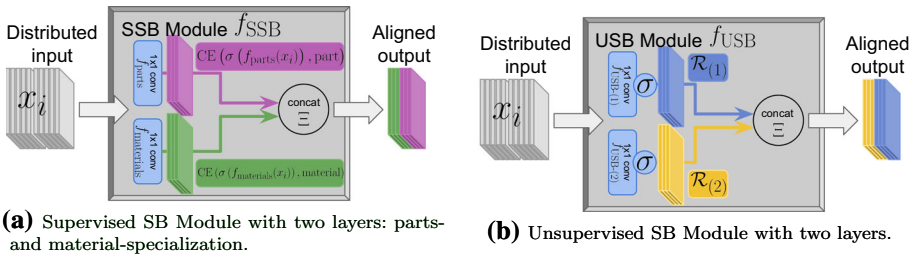
where Ξ defines the concatenation along channels, such that $f_{\text{SSB}} : \mathbb{R}^{N_i \times H \times W} \rightarrow \mathbb{R}^{\sum_{j=1}^{N} N_{\text{cat-}j} \times H \times W}$ for outputs with spatial extent W and H. Recall from our f_{SBNet} definition in Eq. 4 that we need an additional function f_{r} to revert the dimensionality reduction. Instead of adding an additional layer, potentially increasing representational power (not part of our goals), we can also adapt the very first residual block of the subsequent stage $f_{\text{stage-}(i+1)}$ to accept the SSB-output dimensionality. That is, the very first convolutional layer is replaced with a layer working on the SSB dimensionality

Table 1 Selection of 70 Broden-concepts that we deemed relevant for street scene segmentation. The first row lists all used materials, the second row all parts. Each part is organized into part (right) of object/concept (middle)

Materials	Brick, Fabric, Foliage, Glass, Metal, Plastic, Rubber, Skin, Stone, Tile, Wood	
Parts	Sky	Cloud
	Building	Window, Door, Roof, Shop, Wall
	Person	Leg, Head, Torso, Arm Eye, Ear, Nose, Hand, Hair, Mouth, Foot, Eyebrow, Back
	Road	Crosswalk
	Car	Window, Door, Wheel, Headlight, Mirror, Roof, Taillight, Windshield, Bumber
	Van	Window, Door, Wheel, Headlight, Taillight, Windshield
	Truck	Wheel, Windshield
	Bus	Window, Door, Wheel, Headlight, Mirror
	Train	Head, Headlight, Headroof, Coachroof
	Lamp	Arm, Shade, Bulb
	Bike	Wheel, Handle, Saddle, Chain
	Motorbike	Wheel, Headlight, Handle

Table 2 PSPNet-SSB@pyramid trained once with and without softmax activation. With softmax activation performance during inference is impaired substantially.

Softmax	mIoU
Without	74.7%
With	43.0%



(a) Supervised SB Module with two layers: parts- and material-specialization. **(b)** Unsupervised SB Module with two layers.

Fig. 2 Schematics for both Semantic Bottleneck (SB) types. The highly distributed activations x_i from the original layer are fed through one or multiple 1×1-conv layers to specialize channels by semantic alignment. SSBs are concept-supervised and USBs are regularized (via $\mathcal{R}$)

$\mathbb{R}^{\sum_{j=1}^{N} N_{\text{cat-}j} \times H \times W}$. The full SSB-ResNet architecture can be defined as follows:

$$f_{\text{SSB-ResNet}}(x_0) = (f_{\text{stage-}L} \circ \ldots \circ \hat{f}_{\text{stage-}(i+1)} \circ f_{SSB} \circ f_{\text{stage-}(i)} \circ \cdots \circ f_{\text{stage-}1})(x_0), \tag{9}$$

where $\hat{f}_{\text{stage-}(i+1)}$ is the adapted stage. Note, that we do not apply a non-linearity to SSB-outputs. We observed inferior performance during inference when doing so (see Table 2). Nonetheless, softmax (σ) is applied when calculating the cross entropy loss:

$$\mathcal{L}_{\text{cat-}j}(x_i, label) = \text{CE}\left(\sigma\left(f_{\text{cat-}j}(x_i)\right), label\right). \tag{10}$$

Implementation Details For all of our SSBs, we define two bottlenecks: one for parts and one for materials:

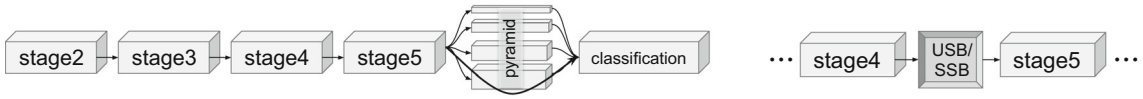
$$f_{\text{SSB}} := \Xi\left(f_{\text{parts}}, f_{\text{materials}}\right). \tag{11}$$

We follow Fig. 2a, which shows the structure of our SSBs. Two linear bottleneck layers (blue rectangles) receive the distributed input x_i from a pretrained model and are supervised by an auxiliary loss to map them to target concepts (colored boxes). Given the dense prediction task of our case study, we use 1×1-convolutions to retain the spatial resolution:

$$f_{\text{parts}}(x_i) = \mathcal{W}_{\text{parts}} \cdot x_i + b_{\text{parts}} \tag{12}$$

$$f_{\text{materials}}(x_i) = \mathcal{W}_{\text{materials}} \cdot x_i + b_{\text{materials}} \tag{13}$$

In our case study, we place SSBs at three different locations in the network. Early, namely after stage3 (see Fig. 3),



(a) Simplified PSPNet architecture indicating stages. Each stage consisting of multiple layers or residual blocks.

(b) PSPNet with SB after stage4.

Fig. 3 Sequence of stages in PSPNet architecture and integration of an SB. MS-OCRNet follows the same structure, but replaces the pyramid and last classification module with a context integration module

Table 3 Segmentation results on Cityscapes validation set for different placements of SSB. ‡ denote results are obtained with models trained from scratch by our setup.

Location	Layer	#concepts (Materials, parts)	mIoU	
			PSPNet‡	MS-OCRNet‡
Vanilla		N/A	77.6	79.6
Early	Stage3 (512 input feat.)	70 (11, 59)	76.4	78.2
Middle	Stage4 (1024 input feat.)	70 (11, 59)	77.4	77.8
Late	stage5 (2048 input feat.)	70 (11, 59)	–	77.3
	Pyramid (4096 input feat.)	70 (11, 59)	74.7	–

middle, namely stage4 and late. Late differs between PSPNet and MS-OCRNet. We apply the SSB after the pyramid module for PSPNet and for MS-OCRNet after stage5.

Training Details The authors description of training the MS-OCRNet is fairly extensive in contrast to PSPNet. It is pretrained on ImageNet as well as Mapillary (Neuhold et al. 2017), utilizes the validation set for training and performs auto-segmentation on the coarse training set to increase the number of images (Tao et al. 2020). Since we are mainly interested in rendering existing networks more inspectable—instead of outperforming baselines –, we choose the training setup of PSPNet to enable comparisons between both models. To construct baselines, we train both models for 200 epochs on the finely annotated Cityscapes data only, with a learning rate of 0.01, weight decay of 5e−4 and "poly" learning rate policy with power 0.9 (Chen et al. 2018; Zhao et al. 2017; Xiao et al. 2018). Integration and training of SSBs is done in two phases. The SSB-ResNet is constructed according to Eqs. 9–12. In the first phase, only the SSB is trained on the out-of-Cityscapes-domain Broden+ by updating the parameters θ_{Phase1}:

$$\theta_{\text{Phase1}} := \Theta\left(f_{\text{SSB}}\right), \tag{14}$$

where Θ returns all SSB parameters (as introduced in Eq. 5). We run the training for 5 epochs with a learning rate of 0.01, weight decay $5e-3$ and the same policy as for the base models. In the second phase, all subsequent layers are finetuned. The parameters subject to optimization are as follows:

$$\theta_{\text{Phase2}} := \bigcup_{k=i+2}^{L} \Theta(f_{\text{stage-}k}) \cup \Theta(\hat{f}_{\text{stage-}(i+1)}), \tag{15}$$

where L defines the number of stages. Phase 2 is trained again on Cityscapes with a lower learning rate of 0.002, keeping the policy and weight decay 5e−4. The number of epochs is fixed to 60. We train both the PSPNet and the MS-OCRNet on 4 GPUs with a total batchsize of 16 and using mixed-precision. The images are cropped for the PSPNet to size 713×713. Our PSPNet achieves 77.6% mIoU on the validation set, on par with the official reported number (78.4%). MS-OCRNet achieves 79.6% mIoU with our training, which is on par with the regular OCRNet performance (also 79.6% Yuan et al. 2020) but not as good as the best reported number (85.1%) involving the full training scheme.

3.1.1 Recovering Performance Using SSBs

As one of our 3 goals for inspectable deep networks, we strive to lose as little performance as possible. We test our SSB-augmented PSP- and MS-OCRNets on the original Cityscapes task and compare mIoUs (see Table 3). We denote an SSB after *stageX* as *SSB@stageX*. We compare the PSPNet integrations first. Given our baseline mIoU of 77.6%, SSB@stage4 is able to recover the full performance, while

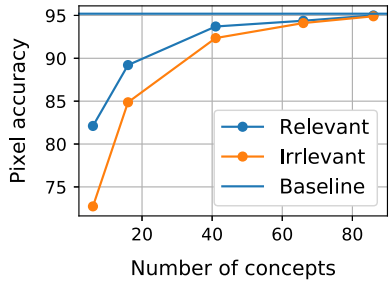


Fig. 4 Task relevant concepts outperform irrelevant ones on PSPNet-SSB@pyramid

the applications to stage3 and the pyramid layer result in a slight decrease (76.4% and 74.7% respectively). Similarly for the MS-OCRNet integrations, we observe only slight performance drops from 79.6% down to 77.3% for stage5. As to be expected, MS-OCRNet achieves overall higher performance on all tested locations. Importantly, for both models: the reduction in the number of channels is substantial (e.g. 1024 reduced to 70 for stage4), indicating room to render complex networks more inspectable. This addresses point 1 of our 3 goals (channel reduction).

Bottleneck Tradeoff Clearly, the performance is strongly dependent on the number of concepts used in the bottleneck. To re-establish the original performance we investigate the idea of increasing the bottleneck size, yet note that increasing the number of concepts is dependent on availability of annotations. In our case, there are no more than our 70 chosen concepts that we deemed necessary for the task. Objects we excluded, that could be considered relevant, are *mirror*, *column*, *stairway* or *door*. Most of these do already exist as parts of objects (see Table 1) or are annotated on very different scenes (e.g. *mirror* is annotated on indoor scenes). In light of our first two goals—dimensionality reduction and semantic alignment—we here choose a minimal set of concepts for our experiments to reduce semantic overlap. To investigate the segmentation performance impact when using over- complete sets as well as reduced sets, we consider the PSPNet- SSB@pyramid configuration and vary the number of concepts from 6 to 86 - selected manually from a set of task relevant concepts and randomly from a disjoint irrelevant set. The set of irrelevant concepts contain all 377 Broden+ concepts minus the 86 relevant ones. We report the pixel accuracy results in Fig. 4 and find that relevant concepts result in improved accuracies over irrelevant ones—at least for small number of concepts.

3.2 Unsupervised Semantic Bottlenecks (USBs)

Clearly, the requirement for additional annotation and uncertainty regarding concept choice is a limitation of SSBs. To address this, we investigate the use of annotation free methods to (i) reduce number of channels, (ii) increase semantic association and (iii) lose as little performance as possible. Similar to SSBs, we address point (i) by integrating layers with low dimensionality. To address (ii) we propose to enforce *non*-distributed representations by performing one-hot relaxation. That is, we strive to achieve one-hot encoded outputs for which only a single channel is active for any given spatial location. We simplify this problem by relaxing an arg max objective and utilize appropriate regularization.

Unsupervised Semantic Alignment One-hot relaxation does not explicitly enforce semantic alignment, yet it adapts a high-level idea implemented implicitly using supervision: each channel should be specialized in representing one concept only. Thus, we strive to maximally specialize each channel via approaching one-hot encodings. As we will later discuss, we find our method resolves implicit semantic alignment well, such that channels represent very specific concepts.

3.2.1 Construction of USBs

As for SSBs, we also integrate the USBs into pretrained models.

Setup Let's specify the USB setup formally for a residual architecture with integration after stage $f_{\text{stage-}i}$. We define the output of stage-i as x_i (see Eq. 2). The composition of the network is similar to the SSB variant (Eq. 9):

$$f_{\text{USB}} := \Xi\left(f_{\text{USB-(1)}}, f_{\text{USB-(2)}}, \ldots, f_{\text{USB-}(N)}\right), \tag{16}$$

for N parallel bottlenecks whose outputs are concatenated along channels via concatenation operator Ξ, such that

$$f_{\text{USB}} : \mathbb{R}^{N_i \times H \times W} \rightarrow \mathbb{R}^{\sum_j N_{\text{USB-}(j)} \times H \times W}. \tag{17}$$

We present an example setup in Fig. 2b which contains two parallel bottlenecks. As for SSBs, we choose 1×1 convolutional layers with bias for each bottleneck $f_{\text{USB-}(j)}$:

$$f_{\text{USB-}(j)}(x_i) := \sigma\left(\mathcal{W}_{\text{USB-}(j)} \cdot x_i + b_{\text{USB-}(j)}\right). \tag{18}$$

Note that, unlike for SSBs, the softmax non-linearity σ is applied both during training and inference as it's integral part to the regularization.

One-hot Relaxation We demand our USBs to produce efficient, sparse codes. By design, we want to have only one active channel per spatial location: That is, $\arg\max_k a_k$, where a_k defines activation of channel k. To relax this non-differentiable problem we utilize softmax and investigate two approaches: adding an additional entropy loss $\mathcal{R}$ or utilizing a parameterization with temperature T. For the first approach using entropy loss, the probability distribution along channels of each $f_{\text{USB-}(j)}$ is used to calculate an additional loss

$\mathcal{R}$:

$$\mathcal{R}_{\text{USB-}(j)}(x_i) = \frac{\lambda}{WH} \sum_{w=1}^{W} \sum_{h=1}^{H} \mathcal{H}\left(f_{\text{USB-}(j)}(x_i)_{h,w}\right), \quad (19)$$

where $\mathcal{H}$ defines the Shannon entropy and x_i is the output of stage-i ($f_{\text{stage-}i}$) resulting in an USB output tensor with $N_{\text{USB-}(j)}$ channels and spatial extent $H \times W$ ($f^{\text{USB-}(j)}(x_i) \in \mathbb{R}^{N_{\text{USB-}(j)} \times H \times W}$). The loss is scaled with factor λ. Here, the entropy $\mathcal{H}$ is calculated along channels $N_{\text{USB-}(j)}$ to measure the uncertainty w.r.t. the active feature for each spatial location h, w. We want to minimize this uncertainty to ideally reach maximum certainty: one-hot encoded outputs. For the second approach, we utilize the softmax temperature parameter, which we anneal during training. Its parametrization can be reduced to the following:

$$\sigma_T(x) := \sigma(x/T). \quad (20)$$

Starting with a high temperature, e.g. $T_0 = 1$ it is reduced quickly in τ training iterations to approach arg max. We define T at timestep t with polynomial decay: $T_t = T_0 + (T_0 - T_\tau) \cdot \left(1 - \frac{t}{\tau}\right)^\gamma$, where γ specifies how quickly T is decaying.

Implementation Details Entropy regularization is scaled with factor $\lambda = 0.1$ while T is kept at 1.0. For annealing we set $T_0 = 1.0$, $T_\tau = 0.01$ and $\gamma = 10.0$ for rapid decay. During inference, we compute arg max **instead of softmax** to acquire one-hot encoded outputs. We choose various bottleneck sizes for our USB experiments to evaluate their impact on performance, discussed in Sect. 3.2.2.

Training Details Since the training is not dependent on additional concept labels, we can train the USB jointly with all subsequent layers. The set of parameters θ subject to training is specified as:

$$\theta := \bigcup_{k=i+2}^{L} \Theta\left(f_{\text{stage-}k}\right) \cup \Theta(\hat{f}_{\text{stage-}(i+1)}) \cup \Theta\left(f_{\text{USB}}\right), \quad (21)$$

where Θ returns all parameters of a function (as introduced in Eq. 5) and L is the number of all stages.

3.2.2 Recovering Performance Using USBs

Here, we show in two parts that first, only USBs trained with temperature annealing produce one-hot encodings and second, that introducing USBs result in little to no performance impact while drastically reducing number of channels. We compare the two regularization methods on two locations of PSPNet (stage3 and stage4) in Table 4.

Temperature Annealing Enables One-hot Encoding Table 4 reports number of channels and two mIoU values per USB configuration—both evaluated on the Cityscapes validation set. The left column reports the unchanged PSPNet-USB performance, and the right (arg max) reports the performance when replacing the softmax with arg max during inference. This comparison enables a simple test. That is, how distributed (or how specialized) are the learned USB-representations. If the representations are not distributed across channels, taking the arg max will induce no or little performance loss. On the other hand, if representations are distributed, a single channel per spatial location does not express all of the required information—the performance will drop. We compare the two different regularization strategies—entropy loss and temperature annealing—and find that only the latter is able to retain performance. On stage3 we see a slight performance drop from 74.7% to 73.3% for 4×50 channels and only a 0.3 percent point drop for stage4 and 2×50 channels. On the other hand, entropy loss does not reduce distributedness sufficiently. The best configuration—stage3 4×50—drops from 74.0% to 36.7%. Thus, the representations of temperature annealed USBs are not distributed across channels.

USBs Reduce Dimensionality Without Performance Impact Next, we investigate the impact of temperature annealing more broadly on multiple layer locations as well as for both models: PSPNet and MS-OCRNet in Table 5. We report dimensions of the bottlenecks, mIoUs after arg max application as well as the number of active channels $\Sigma_{>0}$. The latter derived from an observation that some channels are never active during evaluation (compare columns $\Sigma_{>0}$–active channels–and *#channels*). This resembles results from recent work on differential architecture search also utilizing softmax (Liu et al. 2019; Xie et al. 2019). Overall, this encouragingly indicates further dimensionality reduction. We highlight, that at stage4, stage5 and pyramid it appears that around 20 channels are enough to fully retain the performance—irrespective of the model used. Considering segmentation performance, we observe that only stage3 on PSPNet sees a substantial impact (compare the best 73.3% mIoU versus vanilla PSPNet 77.6%). We conjecture that representations at such an early location in the network are difficult to disentangle, yet we also find USB@stage3 for MS-OCRNet achieves a high 77.4% mIoU. Overall, for MS-OCRNet, we see close to full recovery of mIoU performance across all tested locations.

Parallel Bottlenecks are not Necessary for Good Performance To acquire a better understanding of the impact of using parallel bottlenecks on mIoU performance, we conduct a thorough sweep over USB dimensions and report results for PSPNet in Fig. 5. For each network location, we

Table 4 One-hot relaxation comparison on PSPNet-USBs. To test distributedness, we replace softmax with arg max during inference and compare segmentation performances. Only annealing recovers performance.

Layer	#channels $N \times K$	PSPNet			
		Entropy Loss		Temperature annealing	
		softmax mIoU	arg max mIoU	softmax mIoU	arg max mIoU
Stage3	2×50	73.0	0.1	73.6	**71.7**
	4×50	74.0	36.7	74.7	**73.3**
Stage4	2×10	75.7	32.9	75.1	**75.0**
	2×50	76.0	25.8	75.8	**75.5**

Table 5 Performance comparison between PSPNet and MS-OCRNet after integrating USBs regularized with temperature annealing. $\Sigma_{>0}$ denotes number of active channels—active meaning: the channel has an activation value greater than 0 at least once during validation. Bold numbers highlight best performing configuration per model.

Location	Layer	#channels $N \times K$	PSPNet		MS-OCRNet	
			$\Sigma_{>0}$	mIoU (ss)	$\Sigma_{>0}$	mIoU (ss)
Vanilla		N/A	*all*	77.6	*all*	79.6
Early	Stage3	2×50	33	71.7	17	**76.0**
		4×50	60	**73.3**	48	77.4
Middle	Stage4	1×50	18	72.4	12	75.9
		2×10	16	75.0	11	**77.5**
		2×50	14	**75.5**	20	77.3
Late	Stage5	1×50	–		23	77.2
		2×10			17	77.4
		2×50			23	**78.0**
	Pyramid	1×50	40	**77.5**	–	
		2×10	19	75.4		
		2×50	31	75.6		

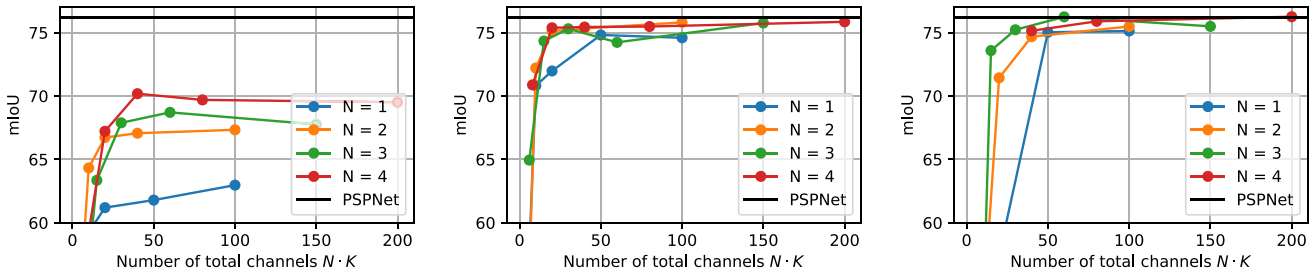
train USBs with varying number of parallel layers N and width K, where $N \in \{1, 2, 3, 4\}$, $K \in \{2, 5, 10, 20, 50\}$. For each N we plot the mIoU over the product $N \cdot K$, stating total number of channels. To decrease training time, we reduce the batchsize from 16 to 4, which decreases PSPNets baseline performance from 77.6% to 76.2% mIoU. Stage4 and pyramid reach baseline results approaching 50 total channels with no substantial gains after. N has the greatest impact on stage3, which reaches highest mIoUs using $N = 3$ (compare $(N = 1, K = 50) \rightarrow 61.8\%$ versus $(N = 3, K = 20) \rightarrow 68.7\%$). Overall, we find that using 3 parallel bottlenecks with $K = 10$ is able to retain performance while having the least total number of channels (discounting active channels $\Sigma_{>0}$). However, even a single bottleneck—resulting in one-hot outputs—with $K = 50$ is able to reach 75% mIoU. We conclude that USBs with parallel bottlenecks ($N > 1$) improve performance quickest, yet $N = 1$ only require few channels more and allow true one-hot encoding.

4 Quantification of Layer Output Inspectability

We present the *Area Under inspectability Curve* (*AUiC*) metric enabling model agnostic benchmarking by measuring alignments between channels and visual concepts. We specify three criteria that AUiC has to satisfy: (i) it must be a scalar measuring the alignment between visual concepts and channel outputs. 0 must indicate no alignment is possible and 1 perfect overlap. (ii) The metric must be model agnostic to allow comparisons between two different activation functions. (iii) The quantification must be computed unbiased w.r.t. the concept area in the image. The fundamental ideas inspiring our metric are based on the frequently cited NetDissect method (Bau et al. 2017). To highlight the differences to our metric we will end this section with a discussion.

4.1 AUiC Metric

Our proposed metric consists of 3 main steps, which we schematically present in Fig. 6. Each channel is consid-



(a) USB@stage3 **(b)** USB@stage4 **(c)** USB@pyramid

Fig. 5 Impact of USB dimensionality trained with temperature annealing on Cityscapes mIoU score for PSPNet. USBs with parallel bottlenecks ($N > 1$) improve performance quickest, yet $N = 1$ only require few channels more and allow true one-hot encoding

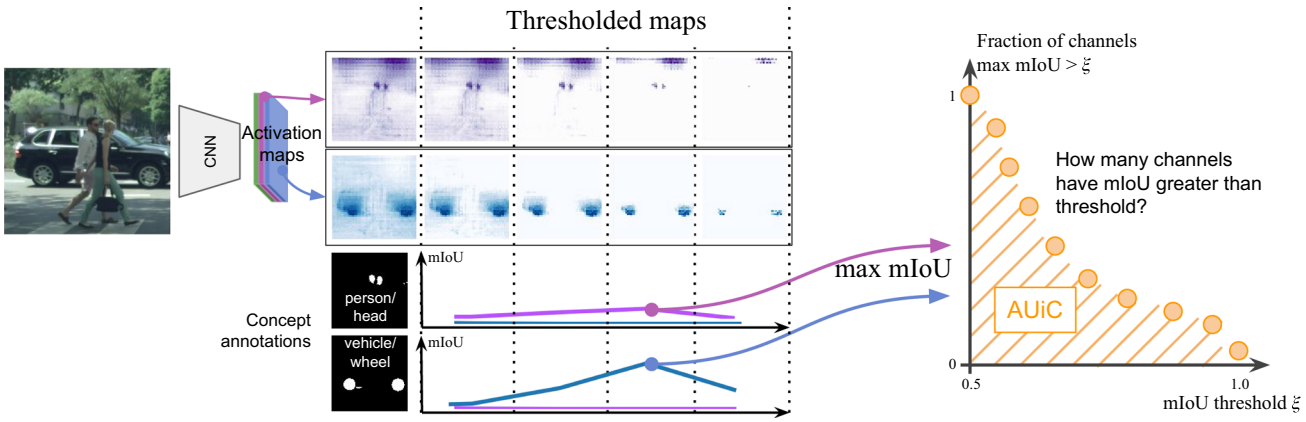


Fig. 6 Schematic calculation of inspectability score AUiC. First, each channel is considered as binary detector—by thresholding its output at various values—and compared with concept annotations (see blue and magenta channel). The best mIoU scores are used to construct the AUiC score. AUiC is derived by counting how many channels have a mIoU greater than a threshold (see plot to the right)

Table 6 Channel identification comparison for PSPNet-SSB@pyramid using either IoU or mIoU. The latter reducing size bias substantially, enabling accurate identifications.

Channel	Trained concept	IoU assignment	Our mIoU assignment
16	Person/hair	torso (0.07) painted (0.06)	person (0.57) hair (0.55)
32	Lamp/shade	painted (0.07) brown (0.06)	shade (0.58) lamp (0.53)
18	Person/foot	torso (0.07) black (0.05)	person (0.53) foot (0.53)
69	Wood material	brown (0.07) painted (0.07)	wood (0.53) floor (0.52)

ered as binary detector by thresholding its output at various values (top center in Fig. 6). For each threshold, the overlap to concept annotations is evaluated via mIoU (bottom center in same figure). Simply speaking, we want to identify the concepts that best describe each channel output. For the final scalar AUiC metric, we take only the best matching channel-concept pair into account and count how many channels in total have a mIoU over a threshold ξ (right in same figure). In the following, we will describe this process more formally.

Channel-concept Matching First, each channel needs to be identified as a detector for a single concept. Given dataset $\mathbf{X}$ containing annotations for concept set C, we compare channel activations A_k and pixel annotations L_c, where $c \in C$. We consider binarized channel outputs M_k via thresholding with θ_k, such that $M_k \equiv M(k, \theta_k) = A_k > \theta_k$ (see thresholded activation maps in Fig. 6). Channel output M_k and concept annotation L_c are compared with IoU: $\text{IoU}(\mathbf{x}) = \frac{|M_k \cap L_c|}{|M_k \cup L_c|}$, $|\cdot|$ denoting cardinality of a set. A few things need to be considered w.r.t. to our criteria. The metric must be unbiased w.r.t. size. IoU penalizes small areas more than large ones, since small annotations are disproportionally more susceptible to noise. Consider an annotation of 2 pixels and one false plus one true positive. The IoU scores 1/3, pulling down the average over all samples. This would become an issue

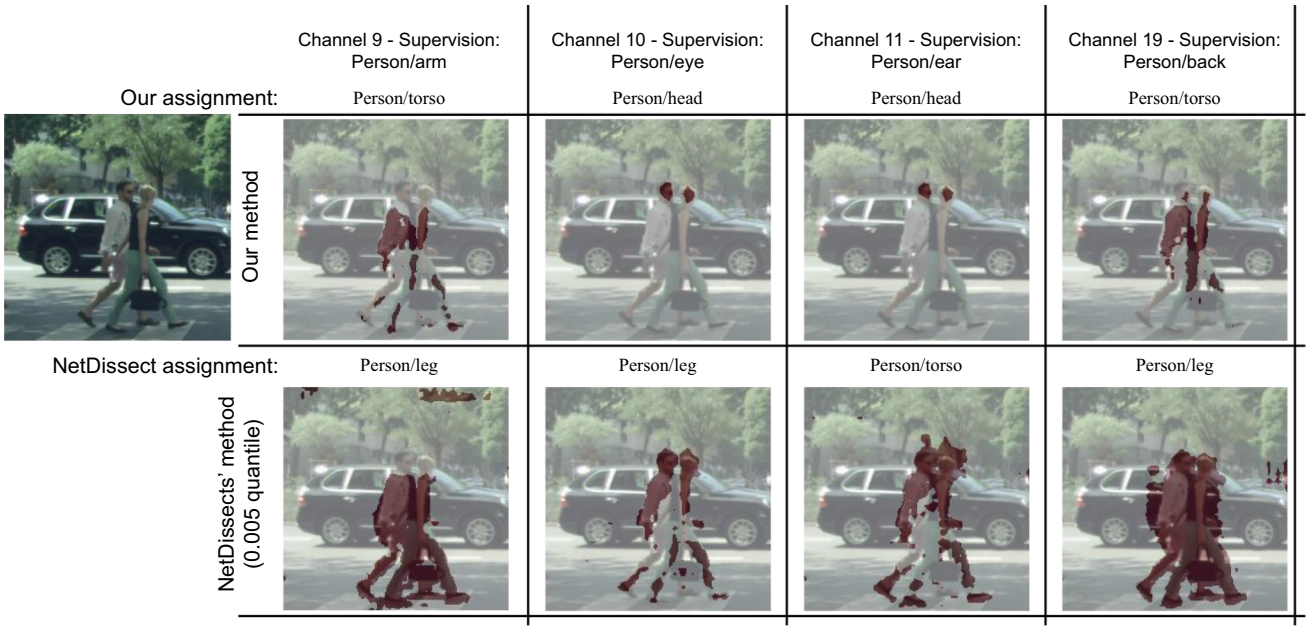


Fig. 7 Comparison of concept assignments between NetDissects method (lower row) and our calibration-free method (upper row) using Cityscapes-Parts. This comparison is performed on the MSOCRNet-SSB@stage5 for which each channel is concept-supervised. The groundtruth concept is stated above each column. Note that our thresholding (heatmap overlay) is better aligned with the actual concept since its threshold determination is an optimization procedure. Both methods have been evaluated on 100 Cityscapes-Parts images

later on when we optimize θ. Here, a bias would lead to wrong identifications. We address this issue using the *mean* IoU of positive *and negative* responses to balance the label area by its complement: $\text{mIoU}(\mathbf{x}) = \frac{1}{2}\left(\frac{|M_k \cap L_c|}{|M_k \cup L_c|} + \frac{|\overline{M_k} \cap \overline{L_c}|}{|\overline{M_k} \cup \overline{L_c}|}\right)$. $\overline{M_k}$ and $\overline{L_c}$ are the complements to the binary activation mask and annotation respectively. The alignment score between channel and concept is subsequently defined over the whole dataset $\mathbf{X}$:

5 Results

$$\text{mIoU}_{k,c}(\mathbf{X}) = \frac{1}{2}\frac{\Sigma_{\mathbf{x}\in\mathbf{X}}|M_k \cap L_c|}{\Sigma_{\mathbf{x}\in\mathbf{X}}|M_k \cup L_c|} + \frac{1}{2}\frac{\Sigma_{\mathbf{x}\in\mathbf{X}}|\overline{M}_k \cap \overline{L}_c|}{\Sigma_{\mathbf{x}\in\mathbf{X}}|\overline{M}_k \cup \overline{L}_c|}. \quad (22)$$

We sum over all samples before computing the fraction to include samples not containing concept c.

So far, the choice of θ_k has been undefined. Yet, the alignment between channel and concept is sensitive to θ_k. We keep the determination of θ_k agnostic to the activation distribution by finding critical point $\theta^*_{k,c}$—now per channel and concept—maximizing $\text{mIoU}_{k,c}(\mathbf{X}, \theta_{k,c})$—now parameterized with the threshold:

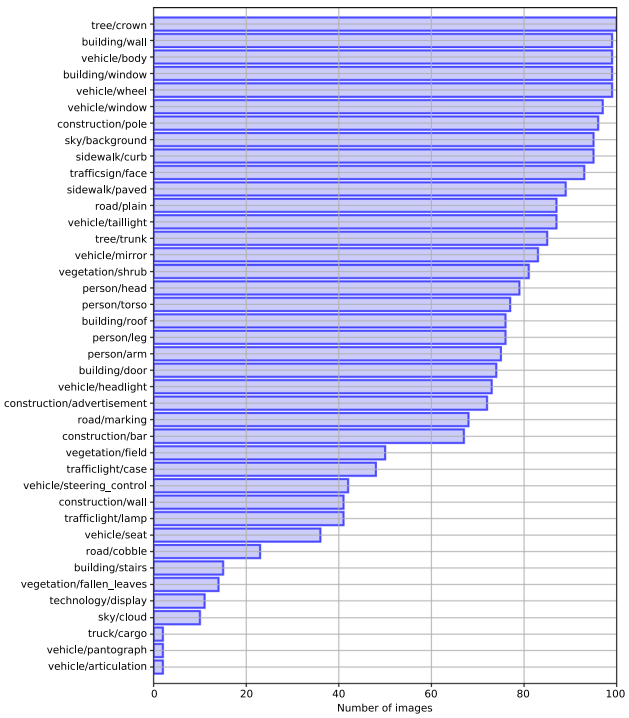


Fig. 8 Number of images containing each concept in Cityscapes-Parts. 37 out of 40 concepts are present in more than 10 out of 100 images

$$\theta^*_{k,c} = \arg\max_{\theta_{k,c}} \text{mIoU}_{k,c}(\mathbf{X}, \theta_{k,c}). \tag{23}$$

An example considering two different concepts (*person/head* and *car/wheel*) is presented in Fig. 6. For each concept, we have identified the threshold $\theta^*_{k,c}$ that maximizes alignment. To assign each channel a single concept, we choose the concept c^* that maximizes mIoU.

$$c^* = \arg\max_c \text{mIoU}_{k,c}(\mathbf{X}, \theta^*_{k,c}). \tag{24}$$

Each concept can be assigned to multiple channels, but not vise versa.

Scalar quantity The second step involves summarizing the identifiability to a scalar value—0 indicating no channel can be identified and 1 all. This is depicted in Fig. 6 to the right. Given a global mIoU threshold ξ we can determine the fraction of channels having a greater mIoU. In order to keep the metric agnostic to the choice of ξ, we define this scalar as the AUC under the indicator function—counting identifiable channels—for all $\xi \in [0.5, 1]$:

$$\text{AUiC} = \frac{2}{K} \int_{0.5}^{1} \sum_{k=1}^{K} \mathbb{1}_{\text{IoU} \geq \xi} \left(\text{mIoU}_{k,c^*} \right) \, d\xi. \tag{25}$$

We coin this scalar AUiC—*Area Under inspectability-Curve*. The factor 2 normalizes the output, such that AUiC ranges from 0 to 1.

Stability w.r.t.$\theta^*_{k,c}$ Since we still choose a single $\theta^*_{k,c}$ to compute our metric, we introduce a second scalar quantity measuring stability when varying $\theta_{k,c}$. For a channel k we retain the selected c^* and marginalize θ out of $\text{mIoU}_{k,c}$. This results in the area under the mIoU curve when varying the threshold:

$$\Delta_{k,c^*} = \int \text{mIoU}_{k,c^*}(\mathbf{X}, \theta) \, d\theta. \tag{26}$$

The ideal inspectable channel consistently responds with perfect overlap *only* to concept c^*. In that case Δ_{k,c^*} will be equal to $\text{mIoU}_{k,c^*}(\mathbf{X}, \theta^*_{k,c^*})$ implying maximal stability. In the general case though, a channel may also respond to other concepts but with smaller activations. This results in an inequality $\Delta_{k,c^*} < \text{mIoU}_{k,c^*}(\mathbf{X}, \theta^*_{k,c^*})$, indicating lower stability. Subsequently, the quotient between these two terms enables the quantification of stability. We define this quantity $\mathcal{S}$ aggregating all channels as the fraction between AUC under Δ and AUiC:

$$\mathcal{S} = \frac{1}{\text{AUiC}} \frac{2}{K} \int_{0.5}^{1} \sum_{k=1}^{K} \mathbb{1}_{\text{IoU} \geq \xi} \left(\Delta_{k,c^*} \right) \, d\xi. \tag{27}$$

5.1 Setup

5.2 Discussion

We conclude by showing that AUiC satisfies our three criteria and contrast it to the NetDissect-measure.

Clear Definition in $[0, 1]$ 0 must indicate no channel alignment—1 perfect alignment for all channels. AUiC satisfies this criterion as it integrates over all mIoU thresholds. NetDissect instead chooses a specific IoU threshold $\xi = 0.04$ that results in fuzzy scores at the bounds. At 1, NetDissects measure gives a false sense of security since all channels only require to pass an IoU of 0.04.

Agnostic to Network Architecture To enable comparison across diverse types of models, we require a metric agnostic to the distribution of the channel outputs. Our AUiC metric satisfies this criterion since it considers the threshold $\theta^*_{k,c}$ that maximizes mIoU. In NetDissects' measure in contrast, the activation binarization threshold θ_k is chosen based on the top quantile level of activations $a_k \in A_k$ such that $P(a_k > \theta_k) = 0.005$. This 0.005 quantile choice is sensitive to the underlying activation distribution, failing for non-Gaussian distributions—e.g. bi-modals and Bernoulli (USBs have 0/1 outputs), for which θ_k could wrongly be set to 1. This results in M_k being always 0. We observe that NetDissect requires further calibration when evaluating our USBs. For example, only 4 out of 25 channels are assigned for PSPNet-USB@stage4 with default settings while ours assigns 20. Additionally, we observe improved assignment results on SSBs. While the outputs here are normal distributed, the optimization procedure to find the best thresholds is able to improve assignments especially for small concept sizes. We show an example in Fig. 7. We choose MS-OCRNet-SSB@stage5 for which all 70 channels are concept supervised and plot 4 different channels with their respective activation maps. The concept assignment is performed by comparing with 40 concepts from our Cityscapes-Parts dataset (introduced in the next section). The first column shows result for channel 9 which represents *Person/arm*. Note that our thresholding results in activation maps that are much closer to the target concept than NetDissects method. Note that many concepts used for supervision (e.g. *eye*, *ear*, *back*) are not present in Cityscapes-Parts. Nonetheless, our algorithm is able to find the closest matches reliably.

Insensitivity to Size of Concept To show size bias using IoU, we conduct a comparison between IoU and mIoU. We compare concept assignments on PSPNet-SSB@pyramid since the channels are supervised and hence pre-assigned. Table 6 presents the assignments of both methods (columns) for 4 channels (rows). mIoU assignments are consistent with the trained concepts, even identifying concept *wood*. Using IoU instead, concepts like *painted*, *black* or *brown* are among

Fig. 9 Examples from our Cityscapes-Parts annotations with 40 different labels focused on parts of Cityscapes objects. 100 randomly selected images from the Cityscapes-dataset have been annotated (excluding test-sets)

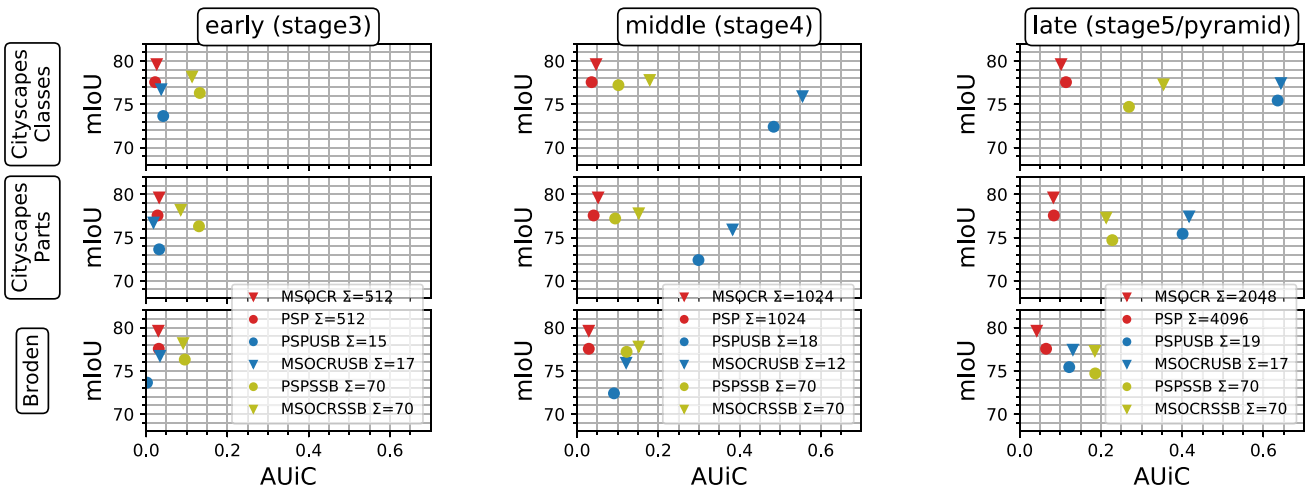


Fig. 10 AUiC–inspectability scores for SSBs (yellow), USBs (blue) and baselines (red). Higher values are better, 1 being perfect channel-concept alignment. SBs substantially improve that alignment and thus: inspectability. Σ indicates number of active channels

Table 7 Averaged stabilities over datasets.

Stability $\mathcal{S}$	Stage3	Stage4	Pyramid
Original	0.034	0.077	0.099
Bottleneck	0.004	0.069	0.082
SSB	0.047	0.099	0.099
USB	0.938	0.945	0.947

the identified. These concepts cover large areas in Broden images making them less susceptible to noise. The average number of pixels per image of *painted* for example is 1087.5, resulting in an IoU of 0.06, while hair has only 93.8 pixels on average and does not show up when using IoU. mIoU on the other hand computes a score for hair of 0.55 for channel 16, which is trained for *hair*. NetDissects metric also utilizes IoU, for which the authors manually adjusted the threshold

to make it unbiased (Zhao et al. 2017). Since this adjustment is done for normal distributions, it's not guaranteed to be unbiased for others.

Section 3 showed drastically reduced channel numbers while retaining performance—achieving goal (i) and (iii). To assess the semantic alignment of channels (goal (ii)) we utilize our AUiC metric to show improved inspectability and highlight qualitative examples in this section.

Datasets We compare alignments with 3 different datasets: *Cityscapes-Classes*, Broden (Bau et al. 2017) and a new dataset: *Cityscapes-Parts*. The broadest dataset we evaluate is Broden which covers 729 various concepts of objects, parts, materials, textures and colors (skipping scene concepts).

Cityscapes-parts Since the Broden images are mostly out of domain w.r.t. Cityscapes, we introduce *Cityscapes-Parts* which includes annotation of subordinate concepts to the 19 Cityscapes classes. The new dataset includes 100 densely annotated images from the Cityscapes dataset covering 40 different concepts that we chose based on the Cityscapes classes. Vehicles including bicycles have been decomposed into wheel, head- and taillight, door, seat, steering control and body. Buildings have not only been decomposed into their direct subordinate concepts like wall and window but also into concepts that often obstruct buildings like advertisements. We also distinguish between plain and cobble road as well as their markings. Three example images are presented in Fig. 9 and concept statistics in Fig. 8. We annotated all images with the labelme toolbox (Wada 2016). We will make all annotations publicly available.

Compared Models From all USB models trained, we select one per network location that strikes the best balance between mIoU and AUiC : namely retaining performance while reducing number of channels. For PSPNet: stage3 - USB2x50, stage4 - USB1x50, pyramid - USB2x10. For MS-OCRNet: stage3 - USB2x50, stage4 - USB1x50, stage4 - USB2x10.

5.3 Quantitative Inspectability Improvements with SBs

We compare vanilla PSPNet/MS-OCRNet with SSBs and USBs and do so for outputs of an early layer: stage3, a middle layer: stage4 and a late layer: pyramid/stage4. AUiCs are collectively shown per layer in three columns in Fig. 10. Each row shows results for a different dataset indicated by the column headers on the left: Cityscapes-Classes, Cityscapes-Parts and Broden. Vanilla network outputs are indicated by color *red*, SSBs by *yellow* and USBs by *blue*. PSPNet and its SB integrations are indicated by circles, MS-OCRNet by triangles.

SSBs Enable Inspection for Subordinate Concepts On each layer and dataset, SSBs outperform the vanilla baselines. Based on the AUiC improvements on the Broden dataset on all layers—on which the SSBs outperform all tested variants—we highlight that it is possible to align intermediate representations with user defined concepts. SSBs improve the AUiCs on Broden from under 0.05 up to 0.2 for the late layers making a big leap forward towards inspectable representations. In comparison to the application of unsupervised USBs though— as we will discuss shortly—large AUiC improvements are challenging to establish as they heavily depend on the choice of concepts. A priori, it is difficult to know which concepts can be aligned well with supervision whilst also allowing for good segmentation performance.

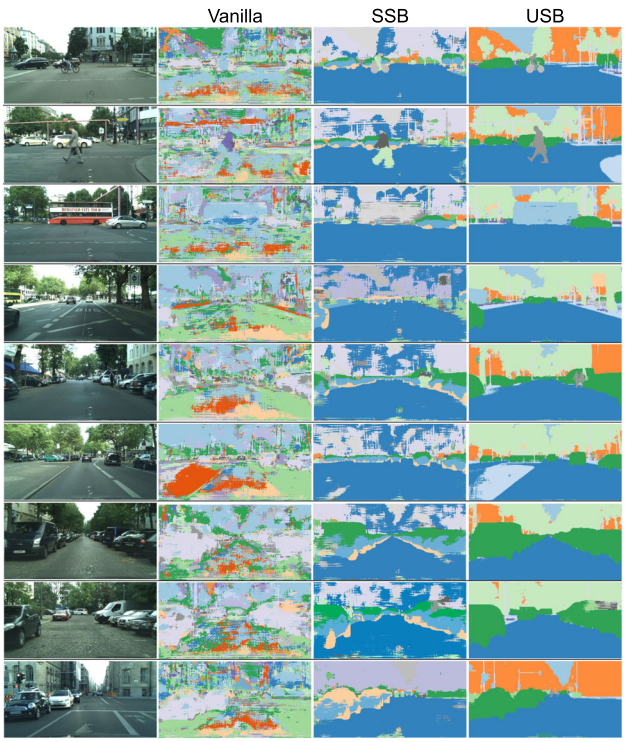


Fig. 11 Top-20 CS-Parts aligned channels only for stage4 from SSB-, USB- and vanilla MS-OCRNet outputs. USBs and SSBs offer better semantic alignment are easier to inspect for concept evidence

USBs Align with Cityscapes-Classes

In comparison to SSBs, USBs align much better with Cityscapes-Classes than subordinate concepts. We see here the greatest increase in AUiC, e.g. from 0.05 to close to 0.6 on stage4 and 0.1 to 0.6 at stage5/pyramid. Clearly, it is much more effective to use unsupervised semantic bottlenecks to improve inspectability, than to force specific concepts onto the network.

USBs Offer High Stability Shown in Table 7, we see USBs offering a clear advantage since their outputs are one-hot encoded: alignments are very stable. SSBs on the other hand report only slight stability improvements over baselines. To answer, whether softmax enables greater stability by default (SSBs have no non-linearity), we measure AUiC and $\mathcal{S}$ for SSB with softmax. Measuring with softmax $T = 1$, we

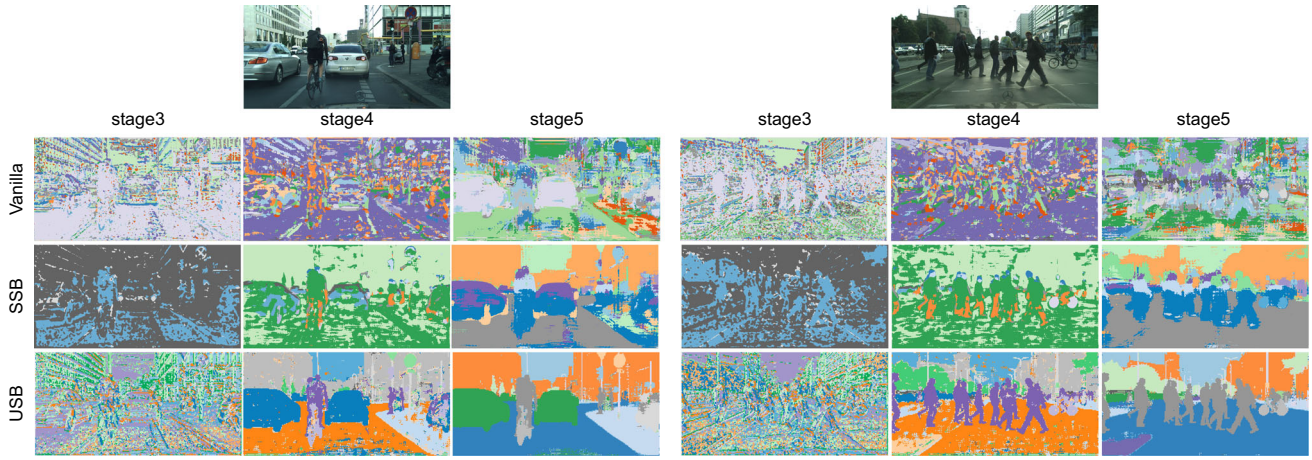


Fig. 12 Top-20 CS-Parts aligned channels from SSB-, USB- and vanilla MS-OCRNet outputs. Each color is mapped to a single output channel. USBs and SSBs offer better semantic alignment are easier to inspect for concept evidence

Fig. 13 Individual channel activations overlaid with the input image of MS-OCRNet-USB@stage5 are coherent and well delineated. Only bright areas are active

find a 2-fold increase of stability to 0.20 but a 3-fold decrease in AUiC to 0.07. While softmax alone increases stability, it does not improve AUiC by default. As noted in Sect. 4, a channel is stable if it responds consistently to the same concept no matter the activation value (arg max USBs have only two states). This is not the case for a regular stage4 and SSB channel, for which the same channel may be active for multiple concepts albeit with low activation. By our definition, this can be inspectable but is not stable. We conclude that the linear SSB-layer is sufficient to align with semantic concepts yet unable to increase stability by a large margin by default. Note that simple bottlenecks show consistently reduced stability (e.g. 0.069 versus 0.077 for bottleneck versus original on stage4).

Representations at Stage3 are Difficult to Align Comparing the AUiC scores between stage3 and other locations, it becomes evident that only SSBs improve inspectability. This indicates an intrinsic difficulty in aligning individual channels with semantics that early and could imply a necessity for distributed representations. We leave this as a challenge for future work.

Conclusion Both SSBs and USBs offer clear advantages over baselines. SSBs are semantically supervised and thus can offer the greatest improvements in AUiC. USBs do not require concept supervision, yet form channels that are well aligned with Cityscapes classes offering a different dimension of inspectability.

5.4 Qualitative Improvements with SBs

To support our quantitative results we compare visualizations of SB-layers and baselines. We show that SB outputs offer substantially improved spatial coherency and consistency.

Top-20 Channels To enable comparison between 1000s and 10s of channels, we utilize the mIoU scoring of our AUiC to rank channels. We show the top-20 channels, assign each a unique color and plot the arg max per location. An inspectable—and thus aligned—channel will result in coherent activations for a unique concept. Visualizations are presented for two images in Fig. 12 for all tested layer locations on MS-OCRNet. 9 additional images are shown in Fig. 11 for stage4 only.

MS-OCRNet Outputs in the first row (*Vanilla*) show the difficulty in interpreting them, since they are highly distributed across channels [also indicated by Fong et al. (2018)]. While still highly irregular in its appearance, activations at stage4 and stage5 show increasing regularities that resemble the Cityscapes classes (e.g. building in purple or cars in light gray).

SSB and USB Outputs Attending to the first image on the left half of Fig. 12, we see spatial coherency greatly improved for USB outputs over baseline. SSBs offer only slight improvements on stage4, but show clear distinctions on stage5 into wheels (beige color for car wheels and light blue for bicycles), car windows (purple), person-legs (dark blue) and torso (light gray). In relation, the USBs appear to form representations that are early aligned with the output classes, which is already evident for USB@stage4. Comparing USB@stage4 and USB@stage5 it appears the semantics of the representations do not differ but are only refined. This gives a clue to the inner workings of deep segmentation nets: the stage outputs get closer to the class labels the higher we go in the layer. Since USBs are unsupervised, they offer an easy access into what concepts have been learned automatically.

Individual USB Channels Finally, we show activation maps of individual USB channels for MS-OCRNet-USB@stage5 in Fig. 13. These convey four points: (i) the maps are visually sparse, (ii) they are spatially coherent and (iii) they are consistently active for the same image concepts. Find the top-20 visualizations, as well as 5 of $\Sigma_{>0} = 17$ channels—one per column. We consistently see the each channel activate for the same concept irrespective of the input sample. The left most channel (11th) activates for road, the next to the right (8th) for buildings and the last three for person, rider and bicyle/motorcycle. We highlight an interesting redundancy in these last three: Person and rider are already distinguished at this stage5 level and kept as individual representations (compare the 3rd and 15th channel in row 3). It appears to be more difficult to determine a pixel belong to person or rider when only concept person and bicycle is retained.

6 Conclusion

In light of the increasing number of data driven models in real world applications like autonomous driving, we demand models to be inspectable at intermediate layers. One issue obstructing inspectability in typical deep networks is that representations are distributed across hundreds of channels and are not semantically aligned (Bau et al. 2017; Fong et al. 2018; Mu and Andreas 2020).

Consequently, in this paper, we proposed supervised and unsupervised Semantic Bottlenecks (SSBs and USBs) to reduce number of channels and align each channel with human interpretable semantics while retaining performance. Additionally, we introduced the AUiC metric quantifying the alignment to enable model agnostic benchmarking as well as stability of same alignment. Using our metric, we showed, that SBs improve baseline scores up to six fold while retaining performance. Overall, we identified USBs to offer greatest inspectability since they are not restricted in the choice of concepts and offer great alignment stability given they are one-hot encoded. While SSBs only increase AUiCs over baselines three fold, they enable the alignment to user-defined semantic concepts.

SBs are easy to integrate into existing architectures and offer great returns in inspectability, making them a useful extension for models used in real world applications.

For future work, we would like to address the following points. First, we want models to have multiple SBs to act as access points to the process pipeline. We conjecture use cases for failure case detection when information along layers indicate conflicting concepts. Secondly, our current methodology involves finetuning the bottleneck as well as all succeeding layers. A combination of USBs with invertible neural networks would spare the need to retrain the target model (Esser et al. 2020) while maintaining inspectability. Finally, we would like to see our semantic bottlenecks applied for other use cases like pathogen segmentation in the medical domain.

Acknowledgements This research was supported, in part, by the Bosch Computer Vision Research Lab Hildesheim, Germany. We thank Dimitrios Bariamis and Oliver Lange for the insightful discussions.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B. (2018). Sanity checks for saliency maps. In *NeurIPS*

Al-Shedivat, M., Dubey, A., Xing, E. P. (2020). Contextual explanation networks. *Journal of Machine Learning Research*, *21*, 194–1

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., Samek, W. (2015). On pixel-wise explanations for non-linear classifier

decisions by layer-wise relevance propagation. *PloS one*, *10*(7), e0130140

Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A. (2017). Network dissection: quantifying interpretability of deep visual representations. In *CVPR*

Bau, D., Zhu, J.Y., Strobelt, H., Zhou, B., Tenenbaum, J.B., Freeman, W.T., Torralba, A. (2019). Gan dissection: visualizing and understanding generative adversarial networks. In *ICLR*

Bell, S., Upchurch, P., Snavely, N., & Bala, K. (2013). Opensurfaces: a richly annotated catalog of surface appearance. *ACM Transactions on Graphics (TOG), 32*(4), 111.

Bucher, M., Herbin, S., Jurie, F. (2018). Semantic bottleneck for computer vision tasks. In *ACCV* (pp. 695–712). Springer

Burgess, C.P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., Lerchner, A. (2019). *Monet: unsupervised scene decomposition and representation*. arXiv preprint arXiv:1901.11390

Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., Su, J.K. (2019). This looks like that: deep learning for interpretable image recognition. In *NeurIPS* (pp. 8930–8941)

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI, 40*(4), 834–848.

Chen, R., Chen, H., Ren, J., Huang, G., Zhang, Q. (2019). Explaining neural networks semantically and quantitatively. In *ICCV* (pp. 9187–9196)

Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., Yuille, A. (2014). Detect what you can: detecting and representing objects using holistic models and body parts. In *CVPR*

Chu, E., Roy, D., Andreas, J. (2020). *Are visual explanations useful? A case study in model-in-the-loop prediction*. arXiv preprint arXiv:2007.12248

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B. (2016) The cityscapes dataset for semantic urban scene understanding. In *CVPR*

Esser, P., Rombach, R., Ommer, B. (2020). A disentangling invertible interpretation network for explaining latent representations. In *CVPR* (pp. 9223–9232)

Fong, R., Patrick, M., Vedaldi, A. (2019). Understanding deep networks via extremal perturbations and smooth masks. In *ICCV*

Fong, R., Vedaldi, A. (2018). Net2vec: quantifying and explaining how concepts are encoded by filters in deep neural networks. In *CVPR* (pp. 8730–8738)

Greff, K., Kaufmann, R.L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. arXiv preprint arXiv:1903.00450

He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., et al. (2017). beta-vae: learning basic visual concepts with a constrained variational framework. *ICLR, 2*(5), 6.

Hooker, S., Erhan, D., Kindermans, P.J., Kim, B. (2019) A benchmark for interpretability methods in deep neural networks. In *Advances in neural information processing systems* (pp. 9737–9748)

Jacobsen, J.H., Smeulders, A.W., Oyallon, E. (2018). i-revnet: deep invertible networks. In *International conference on learning representations (ICLR)*

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. (2018). Interpretability beyond feature attribution: quantitative testing with concept activation vectors (tcav). In *ICML*

Kingma, D.P., Welling, M. (2013). *Auto-encoding variational bayes*. arXiv preprint arXiv:1312.6114

Koh, P.W., Nguyen, T., Tang, Y.S., Mussmann, S., Pierson, E., Kim, B., Liang, P. (2020). Concept bottleneck models. In *ICML*. PMLR

Li, L.J., Su, H., Fei-Fei, L., Xing, E.P. (2010). Object bank: a high-level image representation for scene classification & semantic feature sparsification. In *NeurIPS*

Li, O., Liu, H., Chen, C., Rudin, C. (2018). Deep learning for case-based reasoning through prototypes: a neural network that explains its predictions. In *AAAI*

Lin, D., Shen, X., Lu, C., Jia, J. (2015) Deep lac: deep localization, alignment and classification for fine-grained recognition. In *CVPR* (pp. 1666–1674)

Lipton, Z. C. (2018). The mythos of model interpretability. *Queue, 16*(3), 30.

Liu, H., Simonyan, K., Yang, Y. (2019). DARTS: differentiable architecture search. In *ICLR*

Lundberg, S.M., Lee, S.I. (2017). A unified approach to interpreting model predictions. In *NeurIPS*

Mahendran, A., Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*

Marcos, D., Lobry, S., Tuia, D. (2019). *Semantically interpretable activation maps: what-where-how explanations within cnns*. arXiv preprint arXiv:1909.08442

Melis, D.A., Jaakkola, T. (2018). Towards robust interpretability with self-explaining neural networks. In *NeurIPS*

Mu, J., Andreas, J. (2020). Compositional explanations of neurons. In *NeurIPS*

Neuhold, G., Ollmann, T., Rota Bulo, S., Kontschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV* (pp. 4990–4999)

Petsiuk, V., Das, A., Saenko, K. (2018). Rise: randomized input sampling for explanation of black-box models. In *BMVC*

Ribeiro, M.T., Singh, S., Guestrin, C. (2016). Why should i trust you?: explaining the predictions of any classifier. block In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*

Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.R. (2016). Evaluating the visualization of what a deep neural network has learned. In *IEEE transactions on neural networks and learning systems*

Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D., et al. (2017). Grad-cam: visual explanations from deep networks via gradient-based localization. In *ICCV* (pp. 618–626)

Simonyan, K., Vedaldi, A., Zisserman, A. (2014). Deep inside convolutional networks: visualising image classification models and saliency maps. In *ICLR*

Srinivas, S., Fleuret, F. (2019). Full-gradient representation for neural network visualization. In *Advances in neural information processing systems* (pp. 4124–4133)

Sundararajan, M., Taly, A., Yan, Q. (2017). Axiomatic attribution for deep networks. In *ICML*

Tao, A., Sapra, K., Catanzaro, B. (2020). *Hierarchical multi-scale attention for semantic segmentation*. arXiv preprint arXiv:2005.10821

Wada, K. (2016). labelme: image polygonal annotation with python. https://github.com/wkentaro/labelme

Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J. (2018). Unified perceptual parsing for scene understanding. In *ECCV*

Xie, S., Zheng, H., Liu, C., Lin, L. (2019). SNAS: stochastic neural architecture search. In *ICLR*

Yeh, C.K., Kim, B., Arik, S.O., Li, C.L., Ravikumar, P., Pfister, T. (2019). *On concept-based explanations in deep neural networks*. arXiv preprint arXiv:1910.07969

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H. (2015). *Understanding neural networks through deep visualization*. arXiv:1506.06579

Yuan, Y., Chen, X., Wang, J. (2020). Object-contextual representations for semantic segmentation. In *ECCV*

Zeiler, M.D., Fergus, R. (2014). Visualizing and understanding convolutional networks. In *ECCV*

Zhang, Q., Nian Wu, Y., Zhu, S.C. (2018). Interpretable convolutional neural networks. In *CVPR* (pp. 8827–8836)

Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J. (2017). Pyramid scene parsing network. In *CVPR*

Zhou, B., Bau, D., Oliva, A., Torralba, A. (2017). *Interpreting deep visual representations via network dissection*. arXiv e-prints arXiv:1711.05611

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A. (2017). Scene parsing through ade20k dataset. In *CVPR*

Zintgraf, L.M., Cohen, T.S., Adel, T., Welling, M. (2017). *Visualizing deep neural network decisions: prediction difference analysis*. arXiv:1702.04595