Depth in Convolutional Neural Networks Accomplish Scene Segmentation through the Selection of High-level Object Features

Philip Oosterholt

Brain and Cognitive Sciences

University of Amsterdam

Supervised by Steven Scholte & Noor Seijdel

Faculty of Social and Behavioural Sciences

Brain & Cognition

University of Amsterdam

Date: 01/12/2020

Student number: 10192263

First examiner: Steven Scholte

Second examiner: Ilja Sligte

Abstract

State-of-the-art deep convolutional neural networks (DCNNs) show impressive object recognition capabilities that rival our visual system. The internal state of these networks can predict neural data to an unprecedented degree. For this reason, DCNNs are becoming increasingly popular as computational models for the brain. Recent findings have indicated that DCNNs of sufficient depth can perform implicit scene segmentation through the selection of features belonging to the object regardless of the feedforward architecture. Here, we investigate when and how DCNNs acquire this implicit scene segmentation ability. To this end, we systematically varied the amount of contextual information in the scene and tested feedforward and recurrent networks of various depths. In line with previous results, we found that all networks but especially shallow networks benefitted from contextual information. Moreover, we found that depth helped networks to deal with incongruent contextual information. This ability was not present when the scene reliably predicted the object class, however, when the informativeness of the contextual information decreased, especially the performance of the deeper networks improved quickly. The results indicate that depth enables deep networks to learn implicit scene segmentation. We found no differences between feedforward and recurrent architectures in terms of implicit scene segmentation, indicating that depth through the addition of layers can achieve the same outcome as recurrent processing. Based upon the visualizations of the activation maps we hypothesized that implicit scene segmentation takes place through the selection of more high-level object features relative to low-level background features. We confirmed this hypothesis by selectively removing high-level features of the image and found that the performance of deep networks was disproportionately affected to the degree that shallow networks outperformed deep networks. We conclude that increased network depth allows the network to perform implicit scene segmentation through the selection of high-level features that are unique to the object and absent in the rest of the scene. This process is—at least in outcome—similar to scene segmentation in humans, possibly indicating that scene segmentation in humans does not require explicit mechanisms.

1. Introduction

Despite the enormous amount of variations in object categories and category instances, we can often effortlessly recognize objects within a fraction of a second. Due to the swift nature of this process, some have suggested that *core object recognition*, the part of the processing that is dedicated to recognizing objects, is a feedforward process (e.g. Serre, Oliva & Poggio, 2007). During the feedforward sweep increasingly complex features are extracted within the first 100 to 150-ms (Lamme & Roelfsma, 2000; VanRullen & Thorpe, 2001). These features enable object recognition in scenes that promote the identification of the object. For example, when the object is clearly visible (e.g. not occluded or out of focus), the scene is sparsely populated (e.g. limited number of other objects) and well-organized (e.g. the objects are not cluttered).

Besides object features, features in the background scene can potentially facilitate object recognition. For example, objects placed within congruent scenes that are briefly presented and followed by a mask are reported more accurately and faster than objects placed within incongruent scenes (Davenport & Potter, 2004). However, under challenging circumstances, the feedforward sweep is not sufficient for correctly identifying the object, such as when the object is occluded or embedded in a complex scene (Wyatte, Curan & O'Reilly, 2012; Groen et al., 2018). In this case, the representations derived from the feedforward sweep are likely not sufficiently informative, and there is a need for more complex *visual routines*, such as contour grouping, scene segmentation, and relating this information to knowledge in more abstract memory (Hochstein & Ahissar, 2002; Wyatte et al., 2012; Howe, 2017). These complex visual routines depend on recurrent connections (Lamme & Roelfsma, 2000). For example, V1 neurons initially respond to local image features within their receptive fields. Later, the V1 neurons receive new information via recurrent connections and start to respond to contextual information outside their receptive fields.

In this study, we specifically focus on scene segmentation. Neural correlates of boundary detection and surface segmentation are present in the early visual cortex, but only after the initial feedforward sweep (Scholte, Jolij, Fahrenfort & Lamme, 2008). Even though recurrent processing plays

an important role in scene segmentation, it is still unclear how recurrent processing precisely facilitates scene segmentation. Classical grouping and segmentation theories propose an explicit step where the object is segmented from the rest of the scene and subsequently grouped into one coherent object (Treisman, 1999; Neisser & Becklen, 1975). However, such theories do not elucidate how such processes might be performed by the brain on a computational level. Here, we investigate scene segmentation with the help of computational models.

1.1. Deep Convolutional Neural Networks as Computational Models for the Brain

The understanding of information processing in the brain can be advanced by building computational models that are capable of performing cognitive tasks and consequently explain and predict a variety of brain and behavioural results (Kriegeskorte and Douglas, 2018). Arguably, truly understanding a system necessarily implies that we can build functionally identical computational models. This is still a far cry from reality, however, there is considerable progress towards building models with human capabilities.

The most successful computational models for object recognition in terms of performance and predictive validity are deep convolutional neural networks (DCNNs). The performance of such models rivals that of humans (He, Zhan, Ren & Sun., 2016). Moreover, DCNNs are currently the best predictive computational models for neural data (Cichy & Kaiser, 2019). Later layers in DCNNs can accurately predict activity in visual areas such as V4 and inferior temporal cortex (Yamins et al., 2014; Yamins & DiCarlo, 2016). DCNNs are generally feedforward networks, however, there are recurrent implementations (e.g. CORnet by Kubilius et al., 2018). See Box 1 for more information on DCNNs and their similarities with the brain, Box 2 on DCNN architecture and training, and Box 3 on why DCNNs are powerful models for the brain.

1.2. Scene Segmentation in Deep Convolutional Neural Networks

When investigating object recognition through DCNNs, the question arises whether the networks perform some type of scene segmentation, and if so, how it relates to scene segmentation in humans. DCNNs trained to recognize objects are not explicitly instructed to perform scene segmentation. Importantly, DCNNs do not know beforehand what part of the image is object and scene. It is conceivable that DCNNs do not need to perform any type of scene segmentation at all. Since objects and scenes covary in a meaningful way, DCNNs might opportunistically use every part of the scene for object recognition and make no distinction between object and scene. As it turns out, DCNNs learn, without explicit instructions, at least some form of implicit scene segmentation (Seijdel, Tsakmakidis, De Haan, Bohte & Scholte, 2020). This process is ameliorated through network depth. In the study, Seijdel et al. presented both humans and DCNNs segmented ImageNet objects placed on either congruent, incongruent or no backgrounds at all. Just as humans, DCNNs classified objects more accurately when the background was congruent with the object. Importantly, the difference between objects placed on top of congruent and incongruent backgrounds was more pronounced for shallow networks. Indicating that depth helped to recognize the object despite the incongruent background. Furthermore, the influence of features in the background was more pronounced for shallow networks but almost absent for deeper networks.

While Seijdel et al. demonstrated that sufficient depth enables CNNs to select the relevant object features, it is still unclear when and how the networks learn to acquire this ability. Do DCNNs learn to select features regardless of the amount of contextual information—or does the feature selection strategy depend on the contextual information during training? Moreover, it is still unclear what type of features enable implicit scene segmentation. To answer these questions, we created a Computer Generated Imagery (CGI) dataset in the game engine Unity to control the relationship between the object and background. First, we investigated the effect of contextual information and network depth on object recognition performance by training the networks on objects with various amounts of contextual information. In experiment 1, the networks were trained on various frequencies of

both congruent and incongruent object-scene pairs whereas in experiment 2 the networks were trained on congruent and uninformative object-scene pairs. After training, the networks were tested on congruent and incongruent objects-scene pairs. The two experiments demonstrated that network depth increased the implicit scene segmentation ability and that deep networks can learn the ability even though the contextual effects reliably predict the object class. Next, we used Grad-CAM and showed that network depth increased the use of object features relative to background features. Next, we selectively removed the high-level features from the images and found that now shallow networks outperformed deep networks for the congruent condition. Overall the results indicate that network depth allows for implicit scene segmentation through the selection of high-level features unique to the object and this inadvertently segments the object from the background.

Box 1: Deep convolutional neural networks

What are convolutional neural networks? Convolutional neural networks (CNNs) are neural networks used for analyzing images. The general architecture consists of one input layer for the image, a set of convolutional layers to extract the images features, followed by a fully-connected layer to classify the object. CNNs are primarily feedforward networks. The neurons in each layer are organized in feature maps and each neuron is connected with neurons in the previous layer through a set of weights called filters (LeCun, Bengio & Hinton, 2015). The filters are convolved with the input of the previous layer (the convolutional operation) to see if a feature is present. The filters are shared across the layer and applied across the whole input. Convolutional operations can be followed by a pooling operation. Pooling operations reduce the size of the feature maps to speed up the computations and increase the invariance to small shifts and distortions of the input (Lecun et al., 2015). After each convolutional layer, a nonlinearity is applied to the layer's output. Finally, one or more fully connected layers use the extracted features to classify the object and the corresponding output is translated in probabilities with the help of a logistic function. The weights of the convolutional filters are learned through backpropagation of the error. The error is calculated based on the predicted probabilities and the actual label. With the help of the gradient descent algorithm, the network adjusts its weights so that the error is decreased. Deep convolutional neural networks (DCNNs) are simply CNNs with many convolutional layers.

Similarities between the architecture of the visual cortex and CNNs. CNNs are inspired by the organization of the visual cortex. The combination of convolutional operations, nonlinearities and pooling operations mimic the properties of *simple* and *complex cells* in the visual cortex (LeCun et al., 2015). Similar to the brain, each neuron in CNNs has a corresponding receptive field. The receptive fields in both systems increase in size along the hierarchy of the system. Moreover, both systems process increasingly complex features. Based on these similarities, many neuroscientists argue that DCNNs could function as computational models for biological vision (Kriegeskorte, 2015; Kietzmann, McClure & Kriegeskorte, 2019).

Differences between the architecture of the cortex and CNNs. While CNNs are heavily inspired by the visual cortex the two systems are governed by different constraints. For example, CNNs shared the learned filters across the whole layer while biological neurons can not have access to such non-local information. Moreover,

CNNs are much simpler in terms of connectivity. The brain is abundant with lateral and feedback connections (van Essen & Maunsell, 1983), whereas DCNNs are generally feedforward (there are CNNs with recurrent connections, we discuss these networks later on). Moreover, the artificial neurons are simplified versions of biological neurons (Cichy & Kaiser, 2019). Finally, the implementation of backpropagation is not biologically plausible, in the brain information between synapses can only flow one direction while backpropagation updates the weights of the network in a backward fashion.

Box 2: DCNN Architecture & Training

Architecture. All DCNNs have an input layer, a set of convolutional and pooling operations, and an output layer. Apart from this, the architecture can widely vary. The most obvious difference is the number of convolutional layers. Within each layer, the kernel size of the convolutional operation can vary in size. Additionally, DCNNs vary in the type of non-linear operations, the number and type of pooling operations, and the type of connections between the layers[1]. Since the machine learning field is focussed on maximizing performance, only some of the differences in architecture are relevant for neuroscience. In this study, we focus on the depth of the network and the type of connections (feedforward/recurrent). The role of depth can be researched with the help of ResNets. The number of layers in ResNets can be increased or decreased without alternating the rest of the architecture. We research the role of recurrent connections with the help of CORnet.

Training. Before deployment, DCNNs have to be trained on a dataset with images. Properly selecting training material is crucial for determining the outcome of the properties of the network. This study restricts itself to supervised learning, where DCNNs are presented labelled images for a certain set of times (i.e. epochs). Based on the state of the network predictions for the labels are produced. For each batch of images, the loss is calculated based on the predicted and actual labels. With the help of gradient descent, the weights of the network are adjusted in such a manner that the loss function is minimized. During training, choosing hyperparameters, such as the learning rate, influences the outcome of the final network. Most hyperparameters are less relevant for neuroscience, as many algorithms are not biologically plausible. For performance purposes, DCNNs are initially pre-trained on datasets such as ImageNET, containing 1000 categories with a total of 1.2 million images (Russakovsky et al., 2015). State-of-the-art networks currently reach up to a top-1 accuracy of approximately 88% and top-5 accuracy of 98%[2]. After pre-training, the networks are trained once again on a training dataset that is similar to the real-world data the network needs to predict when employed. By manipulating the training data, we can uncover similarities and differences between DCNNs and humans. For example, Geirhos et al. (2019) showed that ImageNet trained DCNNs are biased for recognizing textures rather than shapes. This discrepancy disappeared when the networks were trained on a stylized version of ImageNet which enhanced shaped-based representations. ImageNet indeed has a strong emphasis on texture, for example, 12% of the categories in ImageNet are different breeds of dogs. Evidently, the strategy to prioritizes texture representations over shape representations is more effective when separating subspecies with similar shapes. This brings us to an important (and often overlooked) property of DCNNs, namely that the dataset is highly influential regarding what type of features and abilities DCNNs learn and use.

[1] This is not an exhaustive list of type of variations

[2] https://paperswithcode.com/sota/image-classification-on-imagenet

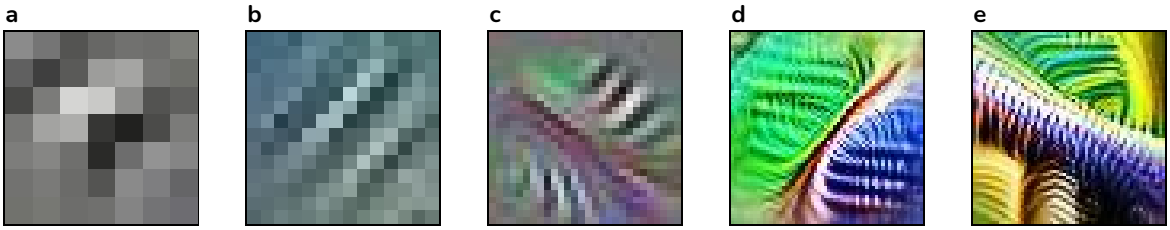
Box 3: Why use DCNNs as computational models of the brain?

Cichy & Kaiser (2019) argue that computational models have two main goals: prediction and explanation of neural processes. Prediction can be seen as both a requirement and stepping stone for explaining brain processes. A model that attempts to explain a phenomenon but is incapable of predicting outcomes is of little value. Despite the differences between the visual cortex and DCNNs, state-of-the-art networks are on par or exceed human performance in object recognition tasks (He et al., 2016). Whereas other more biological constrained models failed to reach human performance levels, e.g. the HMAX model (Riesenhuber & Poggio, 1999). Moreover, DCNNs can predict responses to early visual areas to an unprecedented degree (V1, single-unit: Cadena et al. 2019; Kindel, Christensen & Zylberberg, 2019; V1, fMRI: Zeman, Ritchie, Bracci & de Beeck, 2020; V2: Laskar, Giraldo & Schwartz, 2020, V4, single-unit: Yamins et al. 2014). Importantly, DCNNs are designed to perform object recognition and not to predict neural data per se. While it is not entirely clear why the predictive power of DCNNs is so powerful, the similarity does point to shared computational mechanisms. To make substantiated claims about the computational mechanisms of object recognition the parameters of the networks need to be manipulated systematically. Luckily, DCNNs are the ideal candidates for such studies. Scientists can tweak the architecture and learning rules of the networks with a few lines of code. Moreover, we can present the networks with millions of images within hours. See Box 2 for more information on how DCNNs can be manipulated. Next to the endless manipulation possibilities, we have direct access to the parameters. With techniques such as feature visualization and attribution, we gain unprecedented access to the system. These techniques can be employed as explanations but also as a way to form novel hypotheses and theories.

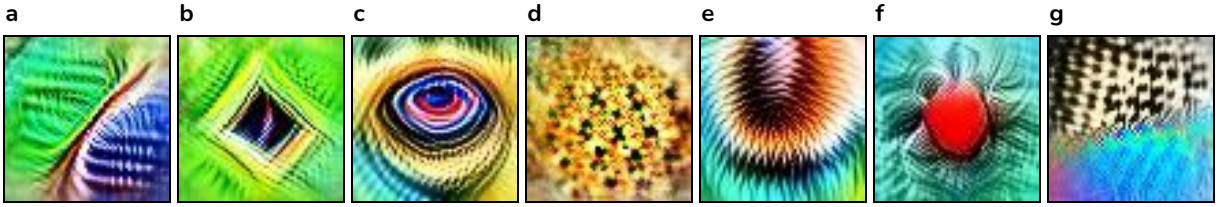
DCNNs can thus become an important tool in the toolbox of neuroscientists. If DCNNs can approximate the performance under various circumstances we can make strong inferences about the computational mechanisms underlying the behaviour and the corresponding neural activity (Scholte, 2018)

Box 4: Low-level vs high-level features

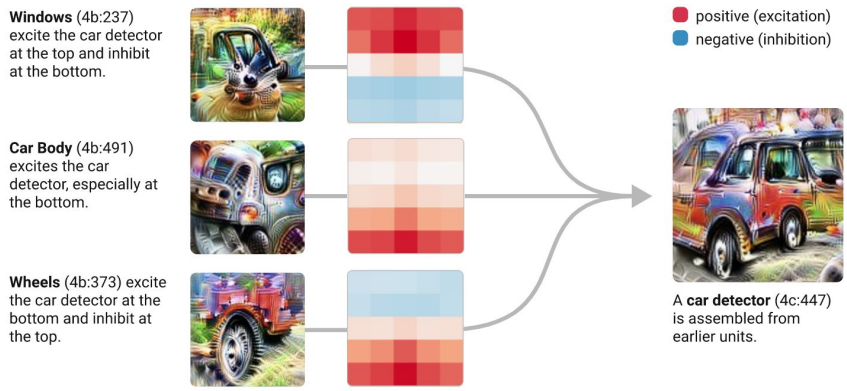
DCNNs can both process low and high-level features in an image. The complexity of the features increases with each convolutional operation, see Box Fig. 1 for the evolution of a boundary detecting neuron. There are various low-level features present in DCNNs, these features can be broadly divided into three categories, shapes, textures and colors (Olah et al. 2020a). Examples of low-level shape features are lines (Box Fig 2a), squares/diamonds (b), circles/eyes (c). Examples of low-level texture features are repeating patterns (d) or fur-like textures (e). Examples of low-level color features are color centre-surround features (f) and color v.s. black and white patterns (g). Most low-level features contain properties of more than one category, for example, neuron 2f uses both color and shape by looking for a certain color surrounded by another color. In later layers, neurons look for features that are more akin to objects (e.g. car windows or wheel) and ultimately can combine specific configurations of features to detect complete objects (Olah et al. 2020b, see Box Fig. 3). The features in early layers are generally easily identifiable whereas in later layers many neurons seem to look for more features at once.



Box Figure 1: Evolution of features throughout the first layers of a DCNN. The figure displays visualizations of what type of stimuli a single neuron fires maximally. These visualizations can be seen as what type of feature detectors neurons are. The method is akin to probing the brain and finding which type of stimuli a neuron (or a brain area) responds to. The main difference between the two methods is that the stimuli of DCNNs are found iteratively by minimizing the loss function. In this figure, five neurons over five different layers of the DCNN InceptionV2 are presented. **A.** Layer I simple Gabor neuron. **B.** Layer II complex Gabor neuron. **C.** Layer III simple line neuron. **D.** Layer IV complex line neuron or an early boundary neuron. **E.** Layer V boundary neuron. Images are adapted from Olah et al. (2020a) under Creative Commons Attribution CC-BY 4.0.



Box Figure 2: Different types of low-level DCNN features. A-C. Shape neurons. **D-E.** Texture neurons. **F-G.** Color neurons. Images are adapted from Olah et al. (2020a) under Creative Commons Attribution CC-BY 4.0.



Box Figure 3: High-level DCNN features. Neurons detecting car parts are combined into a new neuron in the next layer that detects cars with specific configurations. Images are adapted from Olah et al. (2020b) under Creative Commons Attribution CC-BY 4.0.

2. Experiments

2.1. Baseline experiment: Object recognition of segmented objects

Before we started the scene segmentation experiments we assessed the baseline object recognition performance of the networks by training our networks on only the objects (see Fig. 1 for examples). In order to gain systematic control over our training and validation set, we created a dataset by rendering CGI images in Unity. In total there were 26 object categories. The objects were placed in empty (white) scenes and were effectively segmented from the background as only the non-white pixels were part of the object. We pretrained four ResNets of various depths on ImageNet (ResNet-6, 10, 18 and 34).

2.1.1. Methods

Architecture ResNets

For both experiment 1 and 2, we used ResNets of various depths (6, 10, 18 and 34 layers). We opted for the ResNet architecture, considering we can systematically increase or decrease the depth of the networks, while keeping the overall architecture intact (see He et al. 2016 for a detailed explanation). We adapted the ResNet implementation from the PyTorch Torchvision library (https://github.com/pytorch/examples/tree/master/imagenet). Each ResNet contains basic blocks of two convolutional layers, each convolutional layer, consisting of two or more blocks, is followed by a non-linearity (ReLU). ResNet-34 has a total of 16 blocks, ResNet-18 contains 8 blocks, ResNet-10 contains 4 blocks and ResNet-6 contains 2 blocks. Every block contained a skip connection, which allows for increased depth while avoiding vanishing or exploding gradients. Skip connections combine the input from the previous block with the output of the current block at the last convolution. Activation map sizes are decreased through strided convolutions. To increase the stability of the networks, batch normalization is directly applied after each convolution. The last convolution layer is followed by a single fully connected layer of 512 input and 26 output nodes.

Pre-training

The four ResNet networks of various depths were pretrained on the ImageNet database. For preprocessing of the images, we used mean subtraction and division by the standard deviation. A 224 by 224-pixel crop was randomly sampled from an image or its horizontal flip. The validation data consisted of images resized to 256 by 256 pixels and subsequently, centre cropped to 224 by 224 pixels. The initial learning rate was 0.1, thereafter, the learning rate was divided by 10 every 30 epochs. The networks were trained for a total of 150 epochs. Networks were trained using Stochastic Gradient Descent (SGD) with a mini-batch size of 512 per GPU for Resnet-6 & 10, and due to the memory size of our GPU's, a batch size of 384 per GPU for ResNet-18. Weight decay was set to

0.0001 and momentum to 0.9. The networks were pre-trained on multiple Nvidia 1080-ti GPU's. Due to time and computational limitations, we did not train ResNet-34 on ImageNet ourselves, instead, we used the pretrained ResNet-34 available in the PyTorch package.

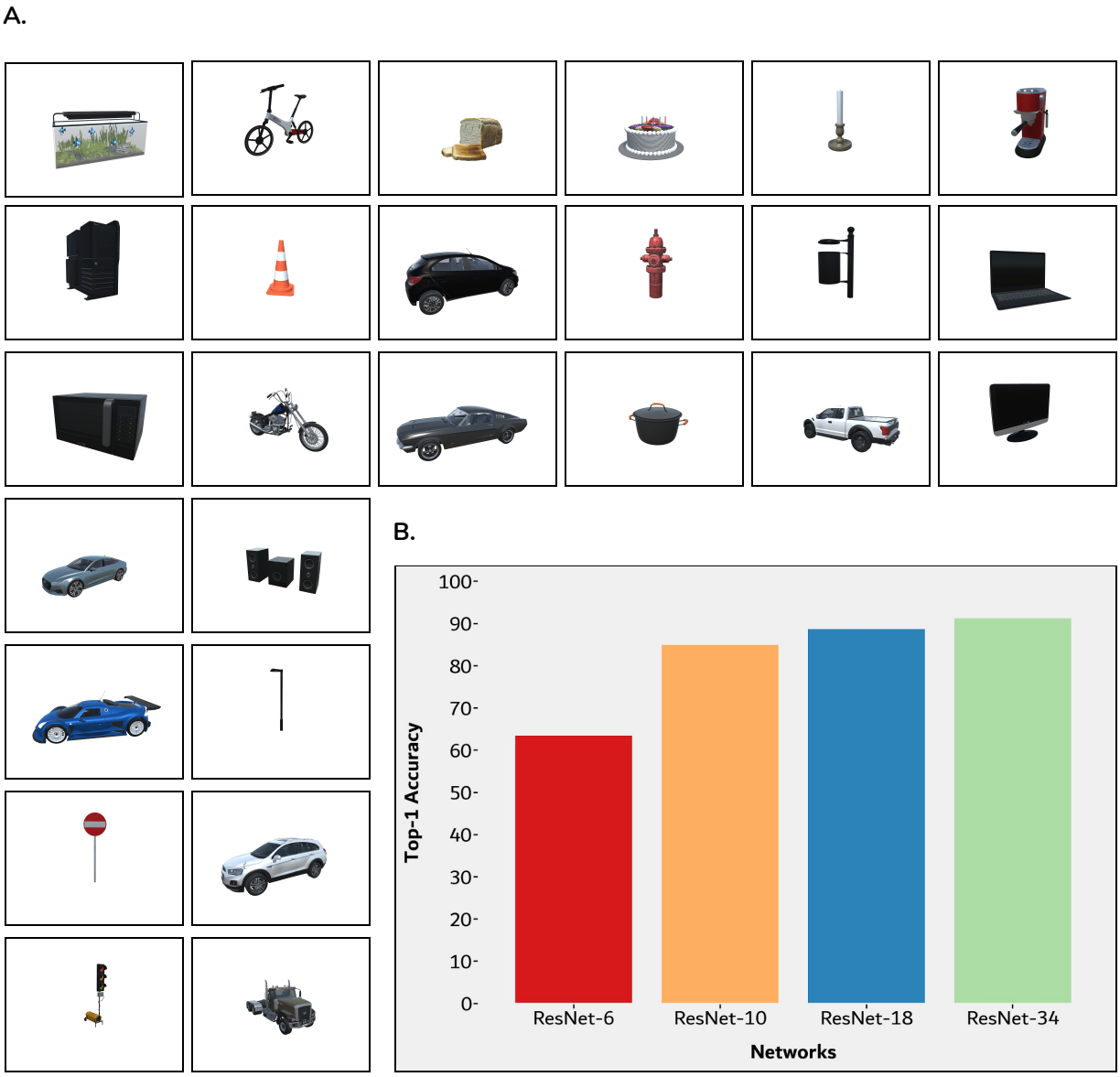


Figure 1: A. Example instances of the 26 object types rendered on a white background. For illustration purposes, we rotated the object and pointed the camera at the object in such a way the object is easily recognizable, in the actual training set this is not always the case. Objects displayed: aquarium, bicycle, bread, cake, candle, coffee machine, computer, cone, hatchback (car), fire hydrant, garbage bin, laptop, microwave, motorbike, muscle car, pan, pick up truck, tv screen, sedan (car), speaker, sports car, street light, street sign, SUV (car), traffic light, truck. **B. ResNet models top-1 accuracy performance on the segmented objects**.

Training stimuli

We created a new database of images using the game engine Unity (version 2018.3). For our 3D Object, we used both native Unity objects and other 3D model formats (e.g. fbx). Materials and textures of non-native Unity objects were manually adapted to reflect the characteristics of the object as closely as possible. In total, the dataset

consisted of the 26 object types. See Fig. 1 for the object categories and examples of the objects. The number of 3D models per category ranged from 20 to 55. The objects were rescaled so that the objects were the same size in length from a fixed camera point. We opted for this approach since rescaling based on width would increase the length of thin objects in a way that only part of the object was visible.

Training set

For the segmented object recognition task, all objects were removed from the scene. The scene only contained the target object, one direct light source and one reflection probe. All images were rendered on top of a white background. The ResNet images were rendered in 2560 by 1440 pixel resolution. We opted for an aspect ratio of 16:9 to include more of the scene in the images. For training, the images were downsampled to 512 by 384 pixels. Initially rendering in a higher resolution allowed us to increase the final quality of the images by avoiding anti-aliasing artefacts. For training we used 500 images per object category, summing up to a total of 13.000 images.

Validation set

Each training set consisted of 75% of the instances of every object type, thereby allocating the remaining 25% to the validation set. Some of the 3D models were slightly different variations of other 3D models (for example, two street lamps with identical style but one had one lightning bulb, the other two). To prevent that the networks might recognize the objects by memorizing the images themselves, similar instances of an object class were always part of the same validation set. The images were rendered and processed in the same manner as the images in the training set. In total, we tested the networks with 150 images for each object category.

Training protocol

The pretrained ImageNet ResNets were used as our training starting point. All networks were trained in the Python PyTorch library. To train the networks on the 26 object categories, the final fully connected layer was removed and replaced by a fully connected layer with 26 output nodes. The training and validation images were normalized by mean subtraction and division by the standard deviation. Since the ResNet architecture allows for variable input sizes we opted for a 4:3 aspect ratio so that the images can contain more contextual information. The input images were 512 by 384 pixels. All layers were fine-tuned during training since this improved the accuracy substantially (this is likely due to the differences between the ImageNet dataset and our CGI dataset). Training started with a learning rate of 0.01, subsequently, the learning rate was divided by 2 every 5 epochs. Networks were trained using Stochastic Gradient Descent with a batch size of 128 for ResNet-6 & 10, 96 for ResNet-18 and 48 for ResNet-34. We trained the network for a total of 20 epochs. Weight decay was set to 0.0001 and momentum to 0.9.

2.1.2. Results

During the testing phase ResNet-6 achieved a top-1 accuracy of 57.23%[3], ResNet-10 84.95%, ResNet-18 88.68% and ResNet-34 91.29%.

2.1.3. Summary and discussion

The results showed that increased depth allowed for better overall performance. The circumstances were of course easy as all features belonged to the object. In the following experiments, we trained and tested the networks on objects placed within congruent and incongruent scenes.

2.2. Experiment 1: Object/scene frequency and scene segmentation

2.2.1. Introduction

In the baseline object recognition task, the ResNets did not perform any form of segmentation. Recognizing objects in real-world situations is far more challenging. One has to select the features that belong to the target object while disregarding irrelevant scene features. When the objects and scenes are covarying systematically (as in the real world), shallow networks might be able to recognize the object by a set of simple features present in both object and scene. However, when the object is incongruent with the scene, the features of the scene are misleading. For those images, the networks have to make a distinction between object and background to correctly classify the object. Do all DCNNs respond in the same way to incongruent contextual information or are deeper networks, by virtue of their increased depth, better equipped for handling these situations? To this end, we placed the 26 object categories in two different types of scenes. Half of the objects were placed in “home” scenes, the other half were placed in “street” scenes. Objects were matched on similarity based on the results of experiment 1 and were placed in opposite conditions. The number of images of congruent object-scene pairs varied from 50 to 100%. See Fig. 2 for a visualization of the experiment. For training on the new dataset, we used the ImageNet pre-trained networks ResNet-6, 10, 18 and 34.

[3] This top-1 score was based upon the same training method as experiment 1 and 2. The performance of ResNet-6 significantly improved to 63.41% when the number of epochs was doubled. The top-1 accuracy for the deeper networks converged before the end of the training and thus did not improve with more training.

Figure 2: Experimental setup experiment 1. The models were trained on objects in two different scene types (home and street scenes). Object categories were assigned a congruent scene type; the congruent scene type is always present in the majority of the training set for a specific object type. In this example, the training set images of the category bread predominantly consisted of street scenes, whereas the training set of the category cake predominantly consisted of home scenes. The number of images of objects in the congruent scene type varied from 50 to 100% (0-50% for the incongruent scene type). The models were tested on novel instances of the object category and scene type. The normal statistical relationship between objects and scenes was disregarded since the models did not have preconceptions about the nature of the relationship.

2.2.2. Methods

Architecture & pretraining

For this experiment, we used the same pretrained ResNets as the baseline task.

Training stimuli

We used the same objects as in the baseline task. Next, we selected two different scene types, namely the interior of a house, a village/city street. All the scenes were lit with daylight lightning (i.e. sunlight). For the city

street, we only used real-time lightning (rendered in real-time). For the house scenes, we used a mix of baked and real-time lightning since scenes lit by real-time lightning looked unrealistic and thus would introduce an artificial difference between the different scene types. Every scene included one or multiple reflection probes. The reflection probe captures a spherical view of its surroundings in all directions. The captured image is then used on objects with reflective materials (e.g. a screen).

Subsequently, we imported all the objects in our scenes. We then predetermined the x, y, z coordinate ranges where we could place our objects in the scenes (without violating physics). With a custom Unity c# script, we randomly placed the objects in the predefined coordinate ranges of the scene. For each image, the camera position was randomly positioned within predetermined ranges. To prevent that the object was always located in the centre, we randomly changed the direction of the camera (though the object was always in view). With this approach, we could create (unlimited) unique images with limited resources. The ResNet images were rendered in the same manner as in the baseline task.

Training set

The objects were initially divided into indoor and outdoor conditions, however, the similarity between the objects within the same condition was high. This resulted in a ceiling effect of the accuracy scores for the incongruent condition due to the lack of contradicting contextual information when classifying two similar objects (e.g. bread and cake). For this reason, we opted to put similar objects into opposing conditions. To determine which object categories were difficult to separate, we constructed a confusion matrix based on the results of the segmented object recognition task. Subsequently, the object pairs that were frequently confused with one another were put into opposite conditions (e.g. bread and cake share similar textures and were hard to pull apart for the networks and thus placed in opposite conditions).

The training set consisted of objects-scene pairs with varying frequencies of specific object/scene pairs. Each dataset contained 500 images per object category, summing up to a total of 13.000 images. The congruent and incongruent conditions are based on which object-scene pairs are the majority of the training set. The number of congruent object-scene pairs varied from 50 to 100% (and the incongruent object-scene pairs 0 to 50%). The frequency of congruent object-scene pairs was fixed per training set and was identical for all objects. For each 1% interval, a unique dataset was created. Two scene types could function as the congruent scene, namely home and street scenes. To create validation sets, the instances of each object category were divided into four separate groups, each group functioned once as the validation set and three times as a part of the training set In total there were 404 unique datasets. See Fig. 2 for a visualization of the experimental set-up and examples of the images.

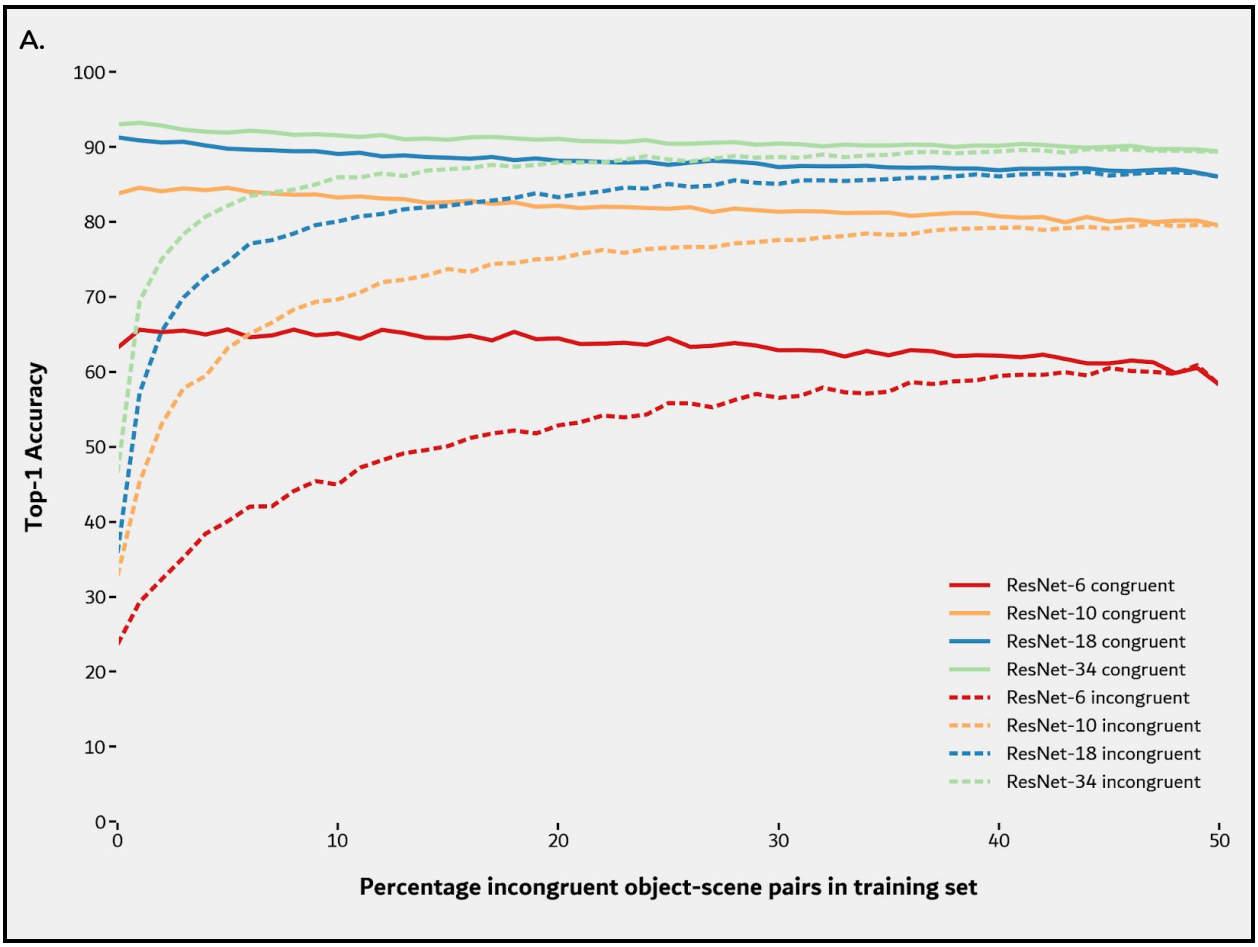
Validation set

We used the same method as the baseline task to construct the validation sets. The validation sets were determined by the training set, for example, when 90% of the object-scene pairs were candle-home, the images of candle-home pairs were congruent and candle-street were incongruent.

2.2.3. Results

The results are displayed in Fig. 3. As expected, all networks performed better on congruent compared to incongruent object-scene pairs. When the training set contained only congruent object-scene pairs, performance on the incongruent object-scene pairs was relatively similar for all networks, ResNet-6 scored a mean top-1 accuracy of 23.64% (SD: 1.75%), ResNet-10 32.77% (SD: 1.54%), ResNet-18 35.85% (SD: 3.00%) and ResNet-34 46.51% (SD: 1.66%). When the number of incongruent object-scene pairs slightly increased, depth helped networks to increase their performance substantially. On the other hand, our shallowest network needed relatively more incongruent counterexamples to increase its performance level. With 10% incongruent scenes present in the training set, ResNet-6 only scored a mean top-1 accuracy of 44.95% (+21.31 percentage points, SD: 2.10%), whereas ResNet-10 scored 69.65% (+36.88 percentage points, SD: 1.91%), ResNet-18 scored 80.04% (+44.19 percentage points, SD: 1.78%) and ResNet-34 scored 85.84% (+39.33 percentage points, SD: 1.92%). We conducted a two-way ANOVA comparing the incongruent top-1 accuracy score of the networks with 0 and 10% incongruent object-scene pairs and found a significant interaction effect between network and percentage ($F(3,54) = 96.47$, $p < 0.001$).

All networks seemed to benefit from congruence between the object and scene as the top-1 congruent object-scene accuracy scores of all the networks was higher than the top-1 scores seen in the segmented baseline task (ResNet-6 +12.55 percentage points, ResNet-10 +3.24, ResNet-18 +4.32 and ResNet-34 +3.06). Moreover, the congruent performance decreased for all networks when the amount of congruent object-scene pairs decreased.



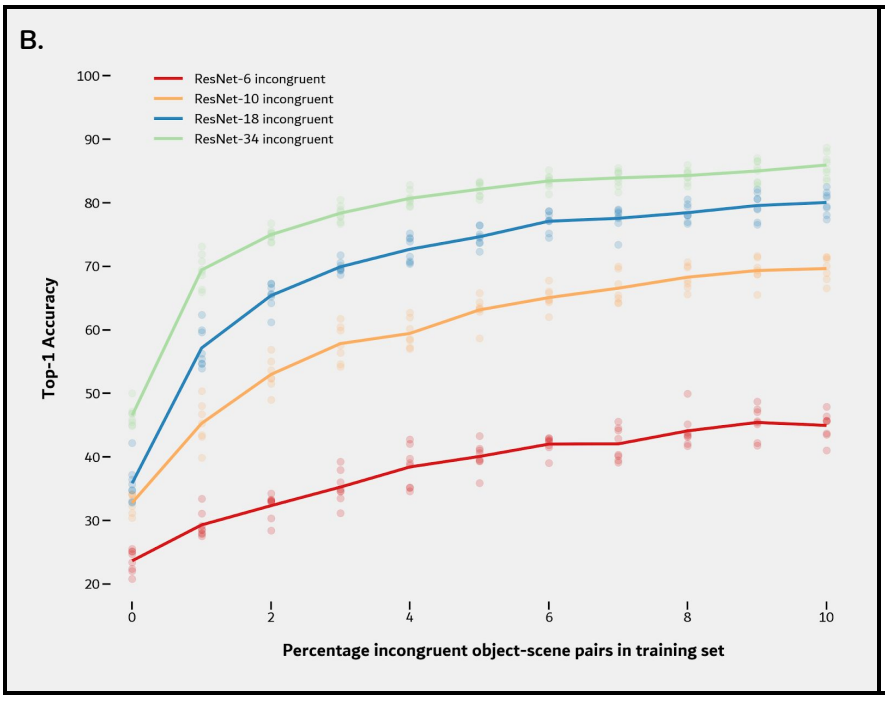


Figure 3: ResNet performance curve experiment 2. The x-axis depicts the percentage of incongruent object-scene pairs in the training set, the y-axis the Top-1 accuracy for the validation set. **A.** Performance curves for both the Top-1 accuracy score for the test images of the congruent and incongruent condition. Note, from 50% and onwards the scene type switches from condition (e.g. data point 0% is equal to 100%). We added each datapoint together and took the average. **B.** Performance curve for the test images of only the incongruent condition for the networks trained on 0 to 10% incongruent object scene pairs. Each dot represents a network trained on a unique training dataset.

2.2.4. Summary and discussion

All networks learned that the scene was extremely reliable for distinguishing different types of objects when there were no incongruent object-scene pairs in the training set. None of the networks learned to focus exclusively on the features belonging to the target object since the networks did not receive an incentive to do so. For this reason, all networks performed well on congruent but poorly on incongruent object-scene pairs. When the number of incongruent object-scene pairs increased, deeper networks quickly improved their ability to recognize objects within incongruent settings. Arguably, the deep networks are capable of discounting the contextual information when seeing examples where the relationship between object and scene is reversed (i.e. incongruent object-scene pairs are *counterexamples* for the congruent object-scene pairs). Moreover, we observed that ResNet-6 benefitted the most from contextual information in the congruent condition. Arguably, shallow networks are less capable of making a distinction between object and the background scene and therefore benefit more from contextual information. Overall, the results imply that increased network depth improves the networks ability to select features belonging to the object and/or ignore features in the scene.

2.2.5. Limitations

While experiment 1 showed clear differences between the networks of various depths the implementation arguably lacks ecological validity. When we observe a specific object in a scene a few times we would automatically classify it as congruent with the scene. The nature of the relationship would not change by seeing the object in other settings (only the strength of the relationship between object and both scene types). In practice, a scene can be either informative for the object class or uninformative. The rapid change in performance for the incongruent condition when the percentage of incongruent object-scene pairs increased could be related to the network classifying the contextual information as unreliable and thus quickly shift towards the use of object features only. While this is still in line with our hypothesis the training itself might give a biased view of how well the networks can

learn to select object features under normal circumstances. In the next experiment, we alter the training set to increase ecological validity.

2.3. Experiment 2: Informativeness of contextual information and scene segmentation (ResNets)

2.3.1. Introduction

The results from the previous experiment showed that depth in DCNNs enabled the networks to either select object features or selectively ignore the scene features when there are incongruent counterexamples present during training. To build a more ecological valid training dataset, we created a new dataset without the counterexamples. To do so, the dataset needs to contain scene types that are congruent to some but not to other objects. Unfortunately, it is infeasible to create a CGI dataset with countless different scene types due to the limited availability. Therefore, we created a third scene type where it was impossible to distinguish objects based on the scene itself. This allowed us to remove the counterexamples from the experiment while still maintaining the contextual information. Additionally, the setup is more realistic as not all scenes help to differentiate between certain objects (e.g. a laptop and computer monitor are both seen in an office setting). See Fig. 4 for a visualization of the experiment. This time, we trained the networks on a subset of congruent and uninformative object-scene pairs. Nature scenes were used as our uninformative scenes. We varied the relative proportions of the two conditions, congruent and uninformative, from 0 to 100%. The networks were tested on the same validation set as the previous experiment, the uninformative scene type was thus not present in the validation set. To avoid confusing, uninformative object-scene pairs are only part of the training. Moreover, incongruent object-scene pairs (i.e. the incongruent condition) are only present during the testing phase. Finally, congruent object-scene pairs are present in the training and testing set. See methods for a more detailed explanation.

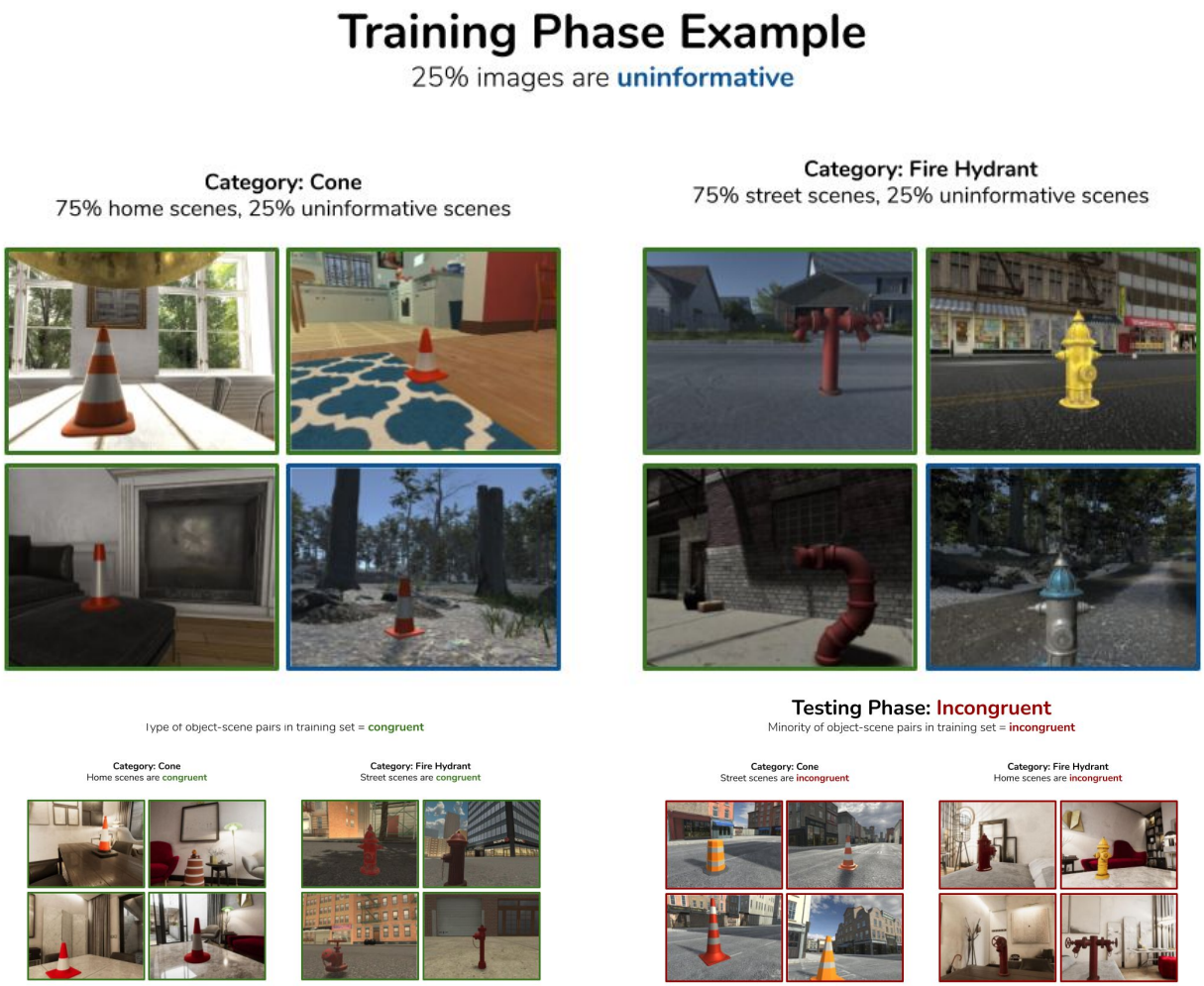


Figure 4: Experimental setup experiment 2. Models were trained on objects within incongruent or uninformative scenes. Incongruent object-scene pairs were thus never present in the training set. The percentage of uninformative scenes varied from 0 to 100%, with intermittent steps of 1%. Models were tested on congruent and incongruent object-scene pairs.

2.3.2. Methods

Architecture & pretraining

For this experiment, we used the same pretrained ResNets as experiment 1.

Training stimuli

We used the same objects as experiment 1. Moreover, we used the same home and street scenes as experiment 1. For the second experiment, we added a third scene type, nature scenes, to the training set. The nature scene was absent from contextual information, rather, the nature scenes were uninformative for the identity of the object.

Training set

Each training set contained 500 images per object category, summing up to a total of 13.000 images. The amount of uninformative object/scene pairs varied from 0 to 100% (and the congruent object-scene pairs as well). The frequency of uninformative object/scene pairs is fixed per training set and is identical for all objects. For each 1% interval, we created a unique dataset. Two scene types could function as the congruent scene (home and street scenes). Just as in experiment 1, the instances of each object category were divided into four separate groups. In total there were 808 training datasets. The validation set was identical to the validation set in experiment 1. Note that for the second experiment the incongruent object-scene pairs were novel whereas for the first experiment the networks had seen the incongruent object-scene pairs in the training set (albeit less often compared to the congruent object-scene pairs).

Validation set

The validation set was identical to the one used in experiment 1.

2.3.3. Results

The results are displayed in Fig. 5. Again, we observed a large difference in performance for the congruent and incongruent condition. However, this time we see a more gradual increase in the performance of the incongruent condition. The difference between the ResNets was again relatively small when there were no uninformative object-scene pairs in the training set. ResNet-6 scored a top-1 accuracy of 24.31% (SD: 2.23%), ResNet-10 32.10% (SD: 2.41%), ResNet-18 36.51% (SD: 2.65%) and ResNet-34 47.45% (SD: 2.53%). The difference between the networks became increasingly larger when the number of uninformative object-scene pairs increased. At 10% uninformative object-scene pairs, ResNet-6 scored a mean top-1 accuracy of 23.42% (-.89 percentage points, SD: 2.06%), whereas ResNet-10 scored 39.8% (+7.7 percentage points, SD: 2.75%), ResNet-18 52.45% (+15.94 percentage points, SD: 2.49%) and ResNet-34 67.2% (+19.74 percentage points, SD: 2.63%). We conducted an ANOVA comparing the incongruent condition top-1 accuracy score of the networks for the training datasets with 0 and 10% uninformative object-scene pairs and found a significant interaction effect between network and percentage ($F(3,56) = 54.84$, $p < 0.001$). As can be observed in Fig. 5b, the incongruent condition performance increased disproportionately for the deeper networks, especially between 0 and 20% uninformative object-scene pairs. Another striking difference is the gap between the individual networks in comparison to the baseline task. For example, the baseline task revealed a

performance difference between ResNet-10 and ResNet-34 of 6.34 percentage points whereas

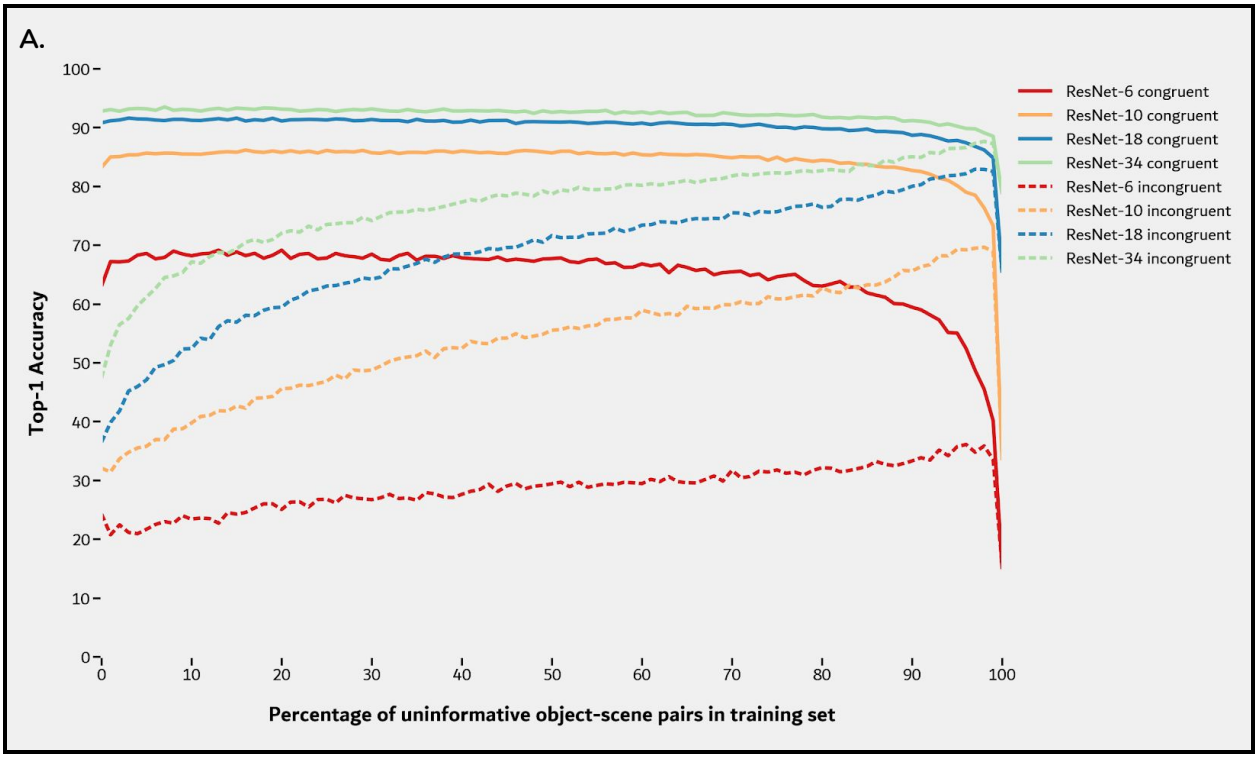


Figure 5a and 5b: ResNet performance curves experiment 2. The x-axis depicts the percentage of uninformative object-scene pairs in the training set, the y-axis the Top-1 accuracy for the validation set. **A.** Performance curves for both the Top-1 accuracy score for the test images of the congruent and incongruent condition. **B.** Performance curve for the test images of only the incongruent condition for the networks trained on 0 to 30% uninformative object-scene pairs. Each dot represents a network trained on a unique training dataset.

the difference is much larger for the incongruent condition of this experiment (at 10% uninformative object-scene pairs the difference is 27.4 percentage points). Moreover, we can see that for the shallow networks, the performance in the congruent condition deteriorated when the number of uninformative object-scene pairs decreased, whereas the performance of the deeper networks remained relatively stable. Interestingly, at 100% uninformative object-scene pairs, there was a performance drop for all networks. This data point was unique since the training set consisted of only one scene type, there is thus no contextual information and less variance present in the training set.

2.3.4. Summary and discussion

We observed that the performance in the incongruent condition of deep networks improved rapidly when the amount of contextual information in the training set decreased. These results can be interpreted in two ways, first of all, the dependence of the deep networks on contextual information is low and the networks, therefore, learn to select object features. On the other hand, deep networks might need fewer data to discover statistical regularities in the data. We argue that the former explanation is most parsimonious with the data. First of all, it is unlikely that the more shallow networks cannot learn the statistical regularities since we can see that the network exploits the statistical regularities when ubiquitous in the training set. Moreover, we selected a training protocol that allows the performance of all networks to converge, more iterations are thus not needed to learn the statistical relationship between object and scene.

Next, we observed that network depth helped networks to deal with objects in novel scenes. The novelty of the scene might make it harder to generalize, however, we can see that especially the shallow networks suffer under these conditions. Whereas humans can, if given enough time, effortlessly segment objects in novel scene types, the shallow networks are less capable of selecting object features in novel scenes. Arguably, the shallow networks confused many previously unseen scene features for object features since the networks did not receive the incentive to decorrelate the scene features with the target object. The drop in performance for ResNet-10 was relatively large because there was more

"room" for the performance to drop, whereas the ResNet-6 was already performing poorly, to begin with. The deeper networks, especially ResNet-34, have less trouble in the novel situation. Conceivably, ResNet-34 learned to select features from the object regardless of the scene, which generalizes to scenes that are completely novel in style and content. The selected object features are arguably complex, these features are inherently less likely to be similar to the novel scene features and thus the deeper networks suffer less from the novel circumstances.

Overall, the results indicate that network depth helps to select features belonging to the target object, especially when the network received an incentive to do so. The incentive is in this case indirect, namely that the scene is not always informative, thus the optimal strategy is to learn features that distinguish object classes from each other. Importantly, all networks receive this incentive, but only deeper networks can adapt their strategy accordingly since the deep networks can leverage high-level features that are only present in the target object.

Next, we discuss the results of the same experiment with the recurrent CORnet architecture.

2.4. Experiment 2: CORnet

2.4.1 Introduction

Using ResNets, our results indicated that depth helped to select features belonging to the object. Possibly, the depth of the networks enables similar computations as the recurrent connections in the brain. If this is the case, the results of adding feedback connections to a DCNN would be effectively the same as adding extra layers to the same network without the recurrent connections. Kubilius et al. (2018) developed a recurrent DCNN, CORnet, to create a network with better anatomical alignment to the brain. We repeated experiment 2 with the three CORnet variants, the feedforward CORnet-Z, the recurrent CORnet-RT and finally CORnet-S, a network with both recurrent and skip connections. See methods for a more detailed description of the architecture.

2.4.2. Methods

Architecture

To investigate the added benefit of recurrent connections, we used three different architectures of the CORnet family. We used the PyTorch implementation of the creators of CORnet (Kubilius et al., 2018, (https://github.com/dicarlolab/CORnet). Each architecture has four areas (i.e. blocks) representing human visual areas V1-V4. CORnet-Z is a simple feedforward network of only one convolutional operation per area. CORnet-RT is a recurrent network with two convolutional operations per area and the recurrent connections are only within the area (no recurrent connections between areas). Finally, CORnet-S is a recurrent network with four convolutional operations per area. Apart from the recurrent connections, each area contains skip connections. Skip connections add the input to the output of one or more convolutional and non-linear operations. Skip connections were introduced in the ResNet architecture and allow for much deeper networks by avoiding the vanishing gradients problem[4] (He et al., 2016). Since the three CORnet architectures have varying depths, a direct comparison between the feedforward and recurrent networks is unfortunately impossible. See Kubilius et al. (2018) for a more extensive description of the CORnet networks.

Training

We used pre-trained versions of CORnet (provided by Kubilius et al., 2018). The final fully connected layer was removed and replaced by one with 26 output nodes instead of 1000 output nodes. As the CORnet architecture does not allow for variable input sizes, we used images of 224 by 224 pixels. The training and validation images were normalized in the same way as during the ImageNet training phase. Training started with a learning rate of 0.02, subsequently, the learning rate was divided by 2 every 5 epochs. Weight decay was set to 0.0001 and momentum to 0.9.

Stimuli, training set and validation set

We used the same approach to create the stimuli and the training/validation set as the ResNets except we rendered the images in a different aspect ratio/resolution since the CORnet architecture did not allow flexible input sizes. The CORnet images were rendered with a 1:1 aspect ratio (rendered in 2560 by 2560 pixels and downsampled to 224 by 224 pixels). The objects in the CORnet images were slightly smaller to offset for the reduced background pixels due to the different aspect ratio.

2.4.3. Results

The results are presented in Fig. 6. The initial differences between the performance of three networks for the incongruent condition were relatively small (CORnet-Z: 34.22% (SD: 4.40%),

[4] The vanishing gradient problem occurs when there are more layers added to the network and as a result the gradients of the loss function approach zero, thereby making it harder or even impossible to continue the training of the network (Hochreiter, Bengio, Frasconi & Schmidhuber, 2001). The ResNet architecture provided a solution to the problem by adding skip connections. These connections add the input of one layer to the output of the next few layers. The skip connections bypass the vanishing gradient problem by sending information from early layers to the deeper parts without the loss of the signal.

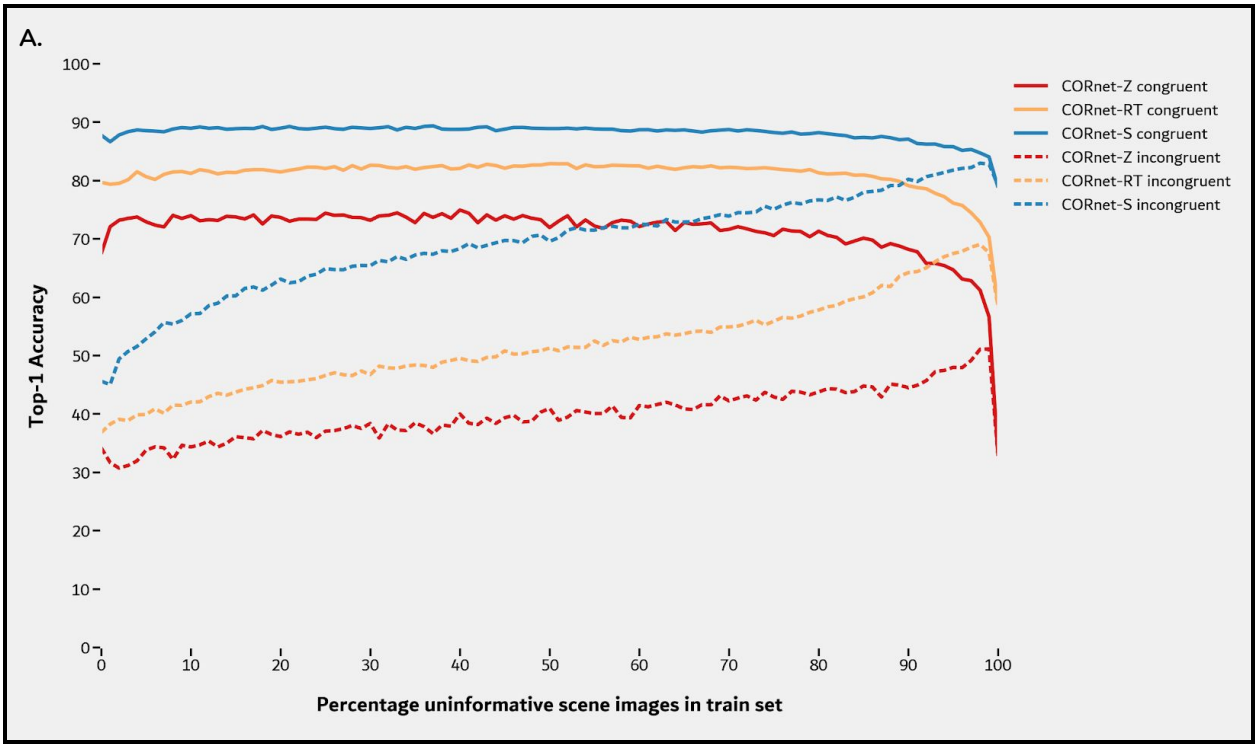


Figure 6a and 6b: CORnet performance curves experiment 2. The x-axis depicts the percentage of uninformative object-scene pairs in the training set, the y-axis the Top-1 accuracy for the validation set. **A.** Performance curves for both the Top-1 accuracy score for the test images of the congruent and incongruent condition. **B.** Performance curve for the test images of only the incongruent condition for the networks trained on 0 to 50% uninformative object-scene pairs. Each dot represents a network trained on a unique training dataset.

CORnet-RT: 36.81% (SD: 2.00%), CORnet-S: 45.62% (SD: 1.81%). As the number of uninformative object-scene pairs increased, the performance for the incongruent condition increased for all networks

but to a lesser degree for CORnet-Z, e.g. at 10%, CORnet-Z: 34.36% (increase of .14 percentage points, SD: 2.36%), CORnet-RT: 42.06% (+5.25 percentage points, SD: 1.50%), CORnet-S: 57.17% (+11.55 percentage points, SD: 1.69%). We conducted an ANOVA comparing the top-1 accuracy score of the networks with 0 and 10% incongruent object-scene pairs and found a significant interaction effect between network and percentage ($F(2,42) = 22.02$, $p < 0.001$).

2.4.4. Comparing CORnet and ResNet

Even though CORnet and ResNet networks were trained on a slightly different aspect ratio we will attempt to draw the comparison between the two network families. Since the networks were trained on different aspect ratios, therefore we opt to report relative performance increases (see methods). Overall, the networks displayed similar trends. Of the three networks, CORnet-Z suffered the most from classifying objects in completely new scenes. The incongruent object-scene performance curve of CORnet-Z is reminiscent of ResNet-6, for example, the performance increase on 0 to 98% incongruent object-scene pairs was 47.58% and 49.29% for ResNet-6 and CORnet-Z respectively. CORnet-RT and CORnet-S fared better as their performance in the incongruent condition improved strongly when the number of uninformative object-scene pairs decreased (e.g. a performance increase on 0 to 98% incongruent object-scene pairs was 87.61% and 81.92% for CORnet-RT and CORnet-S respectively). The relative increase in performance is highly reminiscent of ResNet-10 and ResNet-18, as the performance in both networks increased 117.06% and 127.1%. These results confirm the findings of the previous experiments, for both the ResNets and CORnets network depth helps to select object features while disregarding background scene information.

Since the ResNets and CORnets were trained with a different aspect ratio and resolution we trained ResNet-34 on the same images as CORnet-S. ResNet-34 and CORnet-S are very similar to one another both in size (unrolled depth) and performance (on ImageNet ResNet-34 scored a top-1 accuracy of 73.30% and CORnet-S 74.70%). Based on the experiments with the networks individually, we expect to see similar results apart from a small offset in top-1 accuracy due to the slightly better performance

of CORnet-S overall. As can be seen in Fig. 7, the performance curves of both networks are almost identical. Overall it appears that CORnet-S performed slightly better on congruent object-scene pairs, which was to be expected due to the better ImageNet performance. The performance increase in the incongruent condition is virtually identical up until 80%. We conducted a two-way ANOVA comparing the top-1 accuracy for the incongruent condition and on all uninformative object-scene percentages (in total 1616 data points) and found no interaction effect between model and percentage. Based on these results we argue that the recurrent connections of CORnet-S improve implicit scene segmentation due to the added depth of processing. Apart from the depth of processing, the CORnet-S recurrent connections do not provide any additional benefits.

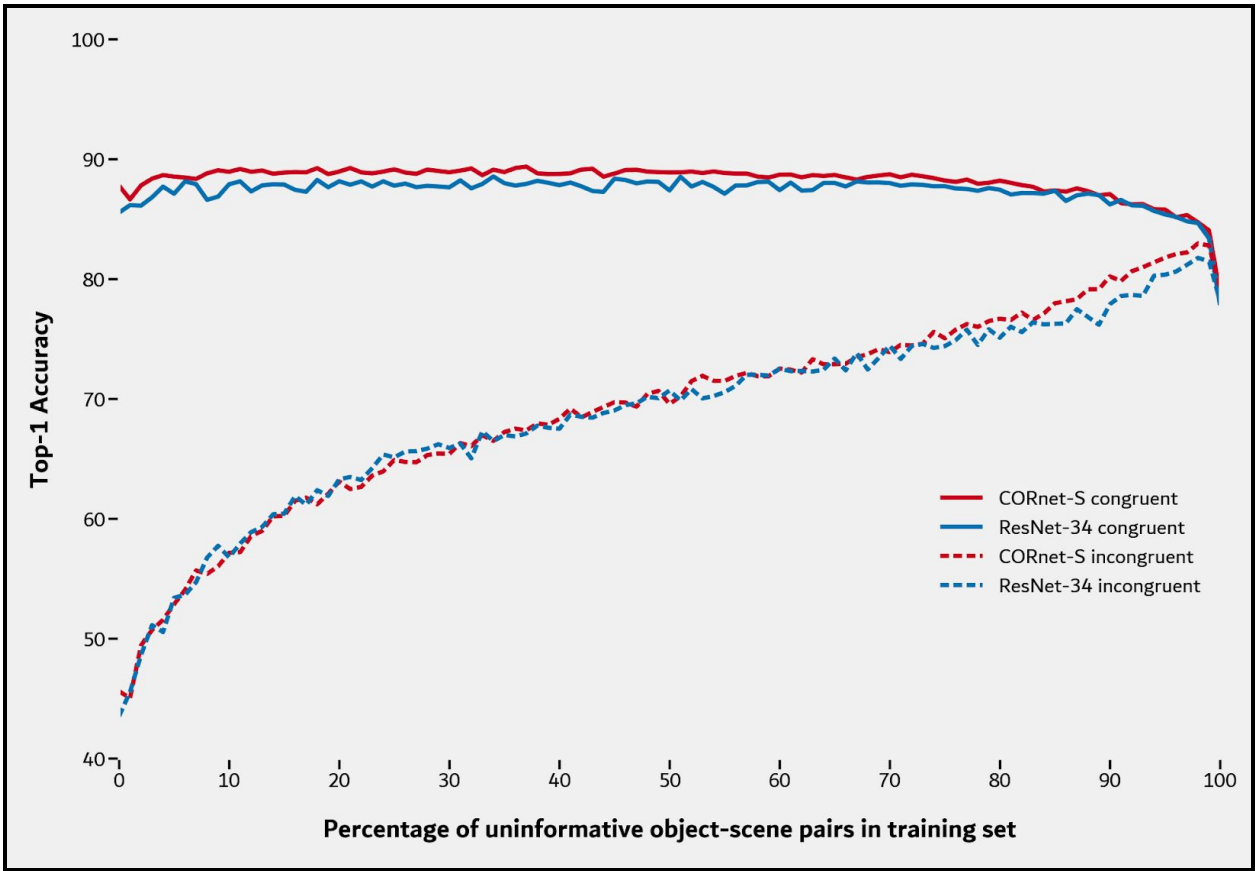


Figure 7: CORnet-S and ResNet-34 performance curves experiment 2. The x-axis depicts the percentage of uninformative object-scene pairs in the training set, the y-axis the Top-1 accuracy for the validation set. Both networks were trained on images of the same aspect ratio and resolution.

2.4.5. Experiment 2 limitations

While experiment 2 improved upon the first experiment there is still a potential problem with the dataset. The deeper networks, especially ResNet-34, might be close to the performance ceiling (ResNet-34 reached 92.27% on average for the congruent condition in experiment 2). The ceiling could reduce the performance differences between networks for both conditions. However, this is not a methodological flaw per se as an increase in both conditions would cancel each other out. Nevertheless, for future studies, it would be interesting to make the task harder. This would also allow testing at what point increased network depth would yield diminishing performance gains. The difficulty of the task could be increased in numerous ways, such as adding extra object and scene types, making the object smaller compared to the background, introducing more variance in the training and validation set. As the addition of 3D models and scenes is labor-intensive we had insufficient time to address this potential issue. Instead, we set out to explore the activation patterns of the networks to corroborate the findings of the previous experiments.

2.5. Visualizing important regions with Grad-CAM

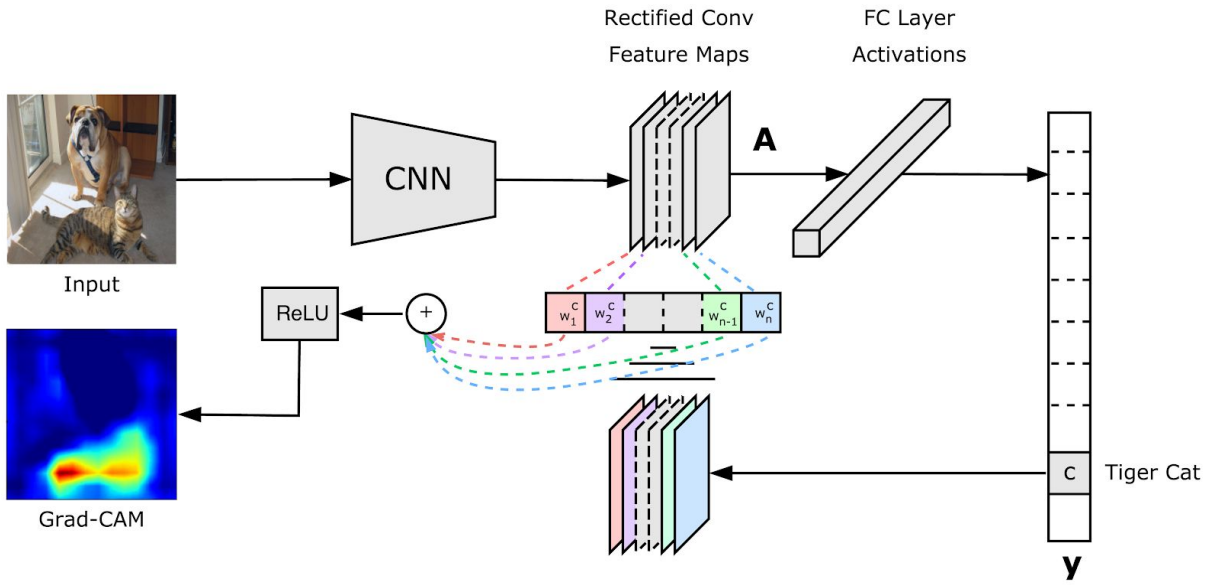
2.5.1. Introduction

Results from the previous experiments suggest that network depths helps to select features belonging to the object while disregarding features in the background. However, at this point, we do not have direct evidence to support the claim since we only know the overall performance and not exactly what drives the differences in performance. To further investigate our hypothesis, we used Gradient-weighted Class Activation Mapping (Grad-CAM, Selvaraju et al., 2016). In short, Grad-CAM shows which parts of the image are used to classify an object by creating saliency maps. Saliency map highlight which pixels of the image were most important for the predicted class (Olah et al., 2018). In Box 5 we discuss Grad-CAM in greater detail. For our experiments, we used Grad-CAM to investigate if network depth influences feature selection strategy. In particular, we look if a network selects features belonging to the object or background. This allows us to see if networks group all relevant features

(both object and scene) together or restrict themselves to features belonging to the object (i.e. no scene segmentation vs. implicit scene segmentation).

Box 5: Grad-CAM: Visual explanations of DCNNs

Grad-CAM is a technique to visualize which parts of the images are used during the classification process. Generally, Grad-CAM is used to see how a network arrives at the predicted class, Grad-CAM saliency maps thus only show the activations for the predicted class. Grad-CAM takes away one of the bigger criticisms towards DCNNs, namely that the networks are black boxes and thus difficult or impossible to interpret.



Box Figure 1: Grad-CAM overview. For each image and each class, Grad-CAM is capable of producing a saliency map. In this case, the image containing both a 'dog' and a 'tiger cat' is used as input. First, the network predicts what (single) object class is most likely the object in the image (in this case 'tiger cat'). Next, we obtain the raw scores of the predicted category preceding the softmax layer. Since we are only interested in one specific class, all the gradients for the other classes are set to zero, and the gradients of the class of interest are set to 1. Then, we compute the gradient of the score for the class of interest (before the softmax) with respect to feature maps of the layer of interest (usually the final convolutional, however in principle we can use any convolutional layer). The obtained gradients are global-average-pooled to obtain the importance weights of our class of interest. Next, we perform a linear weighted combination of activation maps to obtain a coarse activation map. Since we are (generally) only interested in features positively correlated to the class of interest, we apply a rectified linear unit (ReLU) operation to remove negative correlations and get our final saliency map. Figure adapted from Selvaraju et al. (2016) under the Creative Commons Attribution 4.0 International licence.

2.5.2. Methods

Grad-CAM visualization

To analyze which pixels were important for classification we used Grad-CAM (see Selvaraju et al., 2016 for a detailed explanation). Since Grad-CAM visualizes activation maps, there was a difference in activation maps between ResNet-6 and the deeper networks. ResNet-6 has an activation map size of 64 by 48, whereas the deeper networks have an activation map of 16 by 12. The higher resolution of the activation map of ResNet-6 results in

localization of differences in activations with higher precision. This reduced the chance that object-related activation leaked into the background (which can be considered as an artefact to the method, see for example Fig. 8e). For more information on the Grad-CAM method see Box 5.

2.5.3. Results

In Fig. 8 five different Grad-CAM examples are displayed to illustrate the differences between the networks in the use of the object and background features. We restricted ourselves to differences that were repeatedly present across the explored training datasets (with different compositions of uninformative object-scene pairs). The first difference between the networks was the degree to which object and background features were used. An increase in network depth appeared to decrease the use of background features while it increased the use of object features. This was the case for both congruent and incongruent object-scene pairs and was seen in all informative/uninformative object-scene ratios. Furthermore, we observed that increased network depth resulted in the use of larger areas belonging to the target object. Although, not all parts of the objects were prioritized in the classification process. For example, some parts of the object were shared among multiple categories (e.g., street signs, street lamps and traffic lights all share a long pole), these parts were selected to a lesser degree than parts that are unique to the object class. Finally, increased network depth resulted in a more unified region of interest as if the deeper networks saw the target object as one instead of a sum of individual parts.

When only looking at examples we are at risk for an unintended confirmation bias. In the next section, we discuss the quantification of the Grad-CAM method to validate our findings.

Figure 8: Grad-CAM examples. The congruency of the scene with the object is indicated. Wrong predictions are marked red, correct green. The percentages refer to the number of images in the training set with a specific object scene pair (e.g. in the fourth example, in 5% of the training images the traffic light was placed in a street scene and 95% in an uninformative scene, hence the object is incongruent to the training phase). The selected examples show that the shallow networks use more background features whereas deeper networks tend to use more object features (e.g. example A-D). Moreover, the shallowest network at times mistakes background features for object features (e.g. example B & D). Finally, deeper networks use larger areas of the object (or a large part of the object that is distinctly associated with the category).

2.6. Quantifying the Grad-CAM results

2.6.1. Methods object/background ratio

Computing the object/background ratio

We set out to quantify the location of the activations in the saliency map. In particular, we are interested if the activations are localized on the object or background. By quantifying the activations we can see how much object features relative to background features are used by the network (i.e. the object/background ratio).

To compute the object/background ratio we had to process the data to bypass artefacts inherent to the Grad-CAM method. As can be seen in Fig. 8 the deeper networks have a different spatial resolution due to the difference in activation map size. For the deeper networks, this resulted in the spreading of the activations beyond the object itself (e.g. Fig. 8e is a good example). The imprecise localization of the activations is an artefact to the Grad-CAM method. If we in this particular scenario simply ignore the artefact and quantify the results based upon the true labels of the pixels we would heavily bias our results. To fix this issue, we decided to disregard pixels in the near surrounding of the object.

To this end, we created for each test image a mask of the object and the near surrounding area. We first rendered an image (2560 by 1440 pixels) containing both the target object and scene, subsequently, we rendered the image without the object. Whereas normally the object could cast shadows in the scene we disabled this option, otherwise removing the object would result in changes in the lightning of the scene. Next, we used the two images to create a mask by looking for differences between the images with and without the object. The mask was converted to a grayscale image. Identical pixels had a value of 0, other pixels ranged from 0 to 255 (the higher the greater the difference between the same pixel in the two images). Subsequently, we used a convolutional filter with a small kernel size of 10 pixels and selected the pixels which passed a threshold of 35. This allowed us to remove small differences in lightning in the two pictures without affecting the mask of the object itself[5]. Finally, we used a convolutional filter of 30 pixels with a threshold value of .1 to include the near-surround area of the object. We repeated the experiment with a convolutional filter of 10 and 20 pixels. The results were similar as acquired with the filter of 30 pixels. We opted to report the latter since upon visual inspection we found no instances where the object activation leaked into the background. This was not the case for the smaller convolutional filters. The convolutional filter effectively increased the area of the object. This allowed us to make fair comparisons between ResNet-6 and the deeper ResNets since the activation maps of the deeper ResNets were so large that part of the activation in the heatmap was outside the object (see for example the heatmaps of ResNet-18 in Fig. 8)[6]. It's reasonable to assume that this activation was related to the object since the centre of the activation is generally located on the object. Next, we calculated the average activation values (0 to 1) of the pixels belonging to the object and scene. Then, we divided the mean object activation by the mean activation of the scene to create an object-background ratio. The division was done on the average activation over all the images in an object category.

[5] Since Unity is a game engine, all scenes have dynamic elements that change from moment to moment, even though we switched off the animations of nearby objects there were still slight differences in lightning (e.g. incoming light from the windows).

[6] As the activation map sizes of ResNet-6 are larger the network has an advantage over the deeper networks as the chance that an area of the map contains both scene and object is smaller.

We calculated the object/background ratio for validation sets ranging from 0 to 100% with an interval of 10%. Each interval had 4 different validation sets. We only performed the experiment for the ResNets since the CORnet architecture was not compatible with Grad-CAM.

2.6.2. Results object/background ratio

Overall, these anecdotal yet interesting examples reveal that depth enables networks to select features belonging to the target object. To confirm this hypothesis, we quantified the results in two ways. First, we calculated the sum of activation of the area belonging to the object and background, next, we constructed a relative object-background ratio (see methods). See Fig. 9 for the results of each

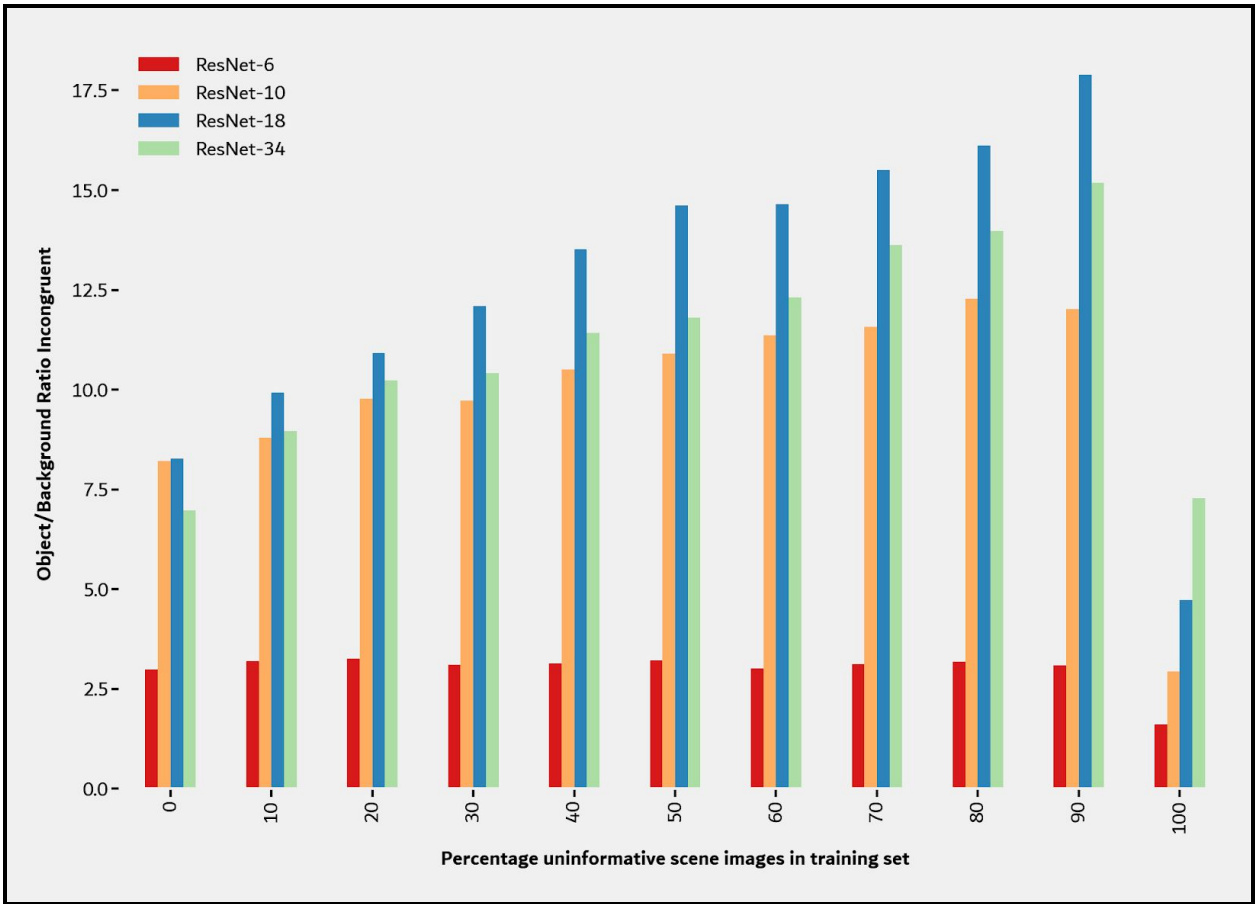


Figure 9: Grad-CAM object/background ratio for the incongruent condition of ResNet-6, 10, 18 & 34. The x-axis depicts the percentage of uninformative object-scene pairs in the training set, the y-axis the activation on pixels belonging to the object divided by background pixels determined by the Grad-CAM method.

network for the incongruent condition. The most striking thing to note is the overall difference of the object/background ratio between the networks, especially when comparing ResNet-6 to the deeper networks. Based upon the object/background ratio of the incongruent condition, we conducted a two-way ANOVA and found a significant main effect for both network ($F(3,132) = 792,46$, $p < 0.001$) and percentage of uninformative object-scene pairs ($F(10,132) = 85,17$, $p < 0.001$). A Tukey's post hoc analysis comparing ResNet-6 with the deeper networks confirmed that the object/background ratio of ResNet-6 was smaller than all other networks (mean difference ResNet-10, 18 and 34 respectively 6.87, 9.59 and 8.12, all $p < 0.001$). Somewhat unexpectedly, the overall object/background ratio for ResNet-18 was consistently larger than ResNet-34. A Tukey's post hoc analysis based upon the two-way ANOVA previously reported confirmed this observation (mean difference 1.46, $p < 0.001$). To date, we do not know exactly why this is the case (see the 2.6.6. for a discussion). On the other hand, the object/background ratio of ResNet-34 was larger than ResNet-10 (mean difference 1.26, $p < 0.001$).

Another crucial difference between the networks is how the object/background ratio evolved when the number of uninformative object-scene pairs increased. We can see that ResNet-6 failed to adapt its feature selection strategy accordingly as the object/background ratio remained unaltered. On the other hand, the deeper networks increasingly favoured the use of object features relative to background features when the contextual information decreased. Moreover, we can see that the object/background ratio of ResNet-18 and 34 appeared to increase faster in comparison with ResNet-10 (from 0 to 90% uninformative object-scene pairs there is an increase of 45.96%, 116.25% and 108.76% for ResNet10, 18 and 34 respectively). To test if there were differences between the feature selection strategy of ResNet-10, 18 and 34, we conducted a two-way ANOVA that examined the effect of network and the percentage of incongruent object-scene pairs ranging from 0 to 90%[7]. The analysis revealed a significant interaction effect between network and percentage, $F(18,90) = 8.47$, $p < 0.001$, implying that the object/background ratio progressed differently for the three networks.

[7] For this ANOVA we omitted ResNet-6, since the inclusion of ResNet-6 will bias the interaction effect between network and percentage of uninformative object scene pairs. We omitted the 100% data point due to the difference between the other data points since it is likely that this datapoint has dynamics not seen in the other data points.

Finally, the object-background ratio for the deepest network had the least trouble with selecting object features in novel scene types.

2.6.3. Discussion object/background ratio

The results indicate that ResNet-6 relied more on background features compared to the deeper networks. Part of the difference can be explained by using more background features for the classification of the object. However, as seen in Fig. 8 (e.g. 8a), ResNet-6 appeared to select background features as if the features belonged to the target object. Arguably, ResNet-6 used low-level features, such as simple shapes and textures, that are ubiquitous in both congruent and incongruent background scenes. Since these features are likely not complex, ResNet-6 failed to make a distinction between what features belonged to the target object and the rest of the scene.

Based on these results we can conclude that network depth enabled the network to adapt its strategy to select object features and disregard background features. This shift in strategy occurs when contextual information is less valuable. Importantly, the strategy is not selected to improve the performance in the incongruent condition since the networks never see incongruent object-scene pairs during training. Rather, for deep networks, the shift in strategy occurs because background features are not the best method for distinguishing different object classes. Instead, selecting high-level features provided a much more reliable strategy.

2.6.4. Results feature selection strategy

When looking at the object/background ratio of each network per condition (see Fig. 10), an interesting difference between the networks emerges. The difference between the object-background ratio of the congruent and incongruent condition remained unaltered for ResNet-6 and 10 until the training dataset contained no uninformative object-scene pairs. In ResNet-18, we see some indication of relative changes between the object-background ratio of both conditions, as the increase in the object-background ratio of the incongruent condition was larger than that of the congruent condition for

the data points 0 to 80% uninformative object-scene pairs. However, the interaction between the two conditions is far from obvious. On the other hand, ResNet-34 appears to have a more clear interaction between the two conditions as the object-background ratio of the incongruent condition increased more (0 to 80%). To confirm this observation, we conducted a two-way ANOVA for each of the networks and found for both ResNet-18 and 34 an interaction effect between percentage of uninformative object-scene pairs and condition (respectively $F(9,60) = 4.07$, $p < 0.001$ and $F(9,60) = 4.83$, $p < 0.001$).

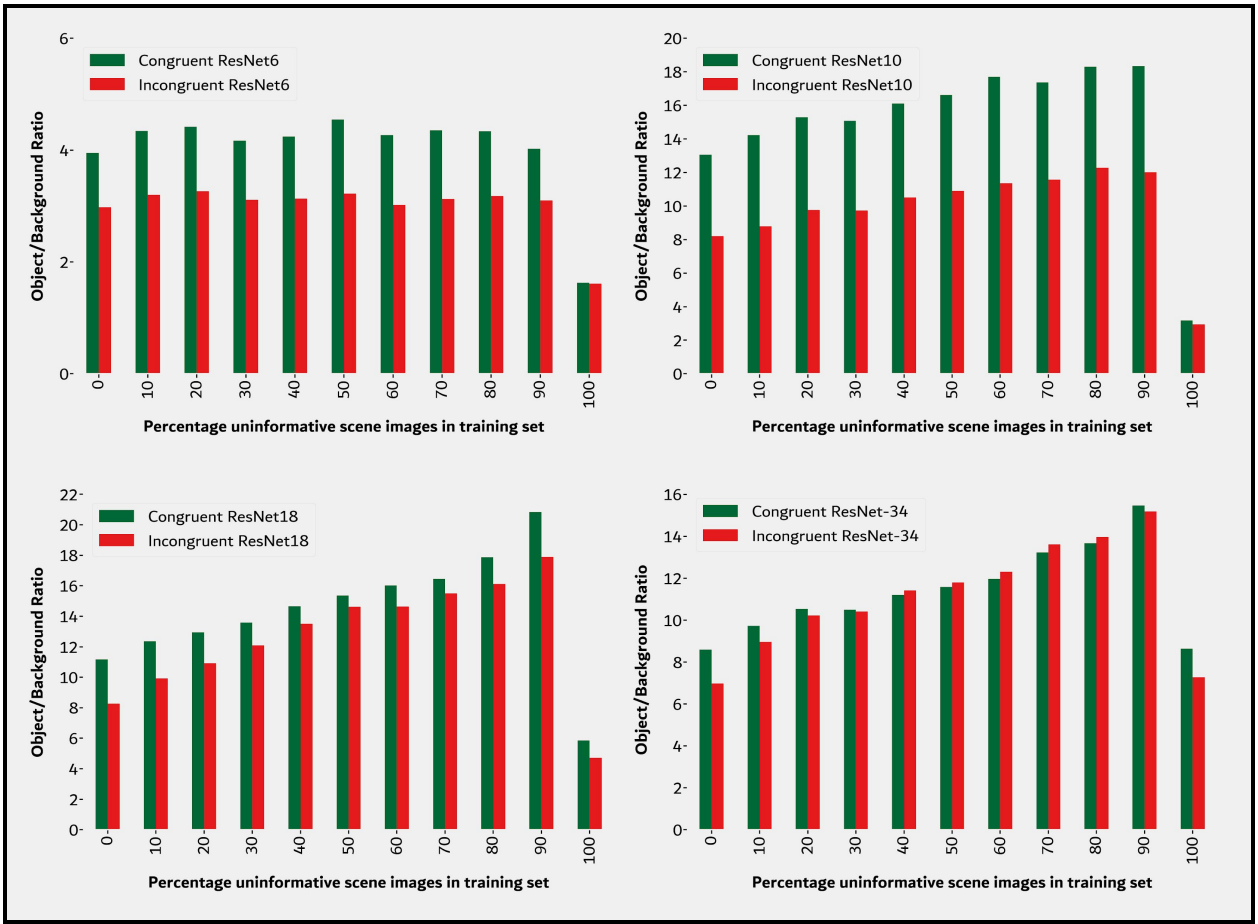


Figure 10: Grad-CAM object/background ratio for the both conditions of ResNet-6, 10, 18 & 34. The x-axis depicts the percentage of uninformative object-scene pairs in the training set, the y-axis the activation on pixels belonging to the object divided by background pixels determined by the Grad-CAM method.

2.6.5. Discussion feature selection strategy

Network depth might enable deep networks to selectively adapt the feature selection strategy per case. While the following is still speculative, deep networks might be able to increase the use of

object features relative to background features when there is conflicting information (i.e. the incongruent object-scene pairs). Based upon this data alone we can not argue for certain that this is the case therefore new experiments should verify the hypotheses.

2.6.6. Limitations: Inconsistent Grad-CAM findings

The performance curves of ResNet-18 and 34 show clear differences that support our main conclusions, however, the Grad-CAM results are somewhat inconsistent with the overall performance. The overall object/background ratio was higher for ResNet-18 than ResNet-34. We investigated if the differences arose from a systematic bias but did not find any differences between the network. We think the most likely explanation of the difference is the pixel-based approach of the Grad-CAM and especially the fact that the magnitude of the contribution of each part of the image is normalized. Due to the omission of this information, we might have an area that contributes 10 times as much as another area, yet the normalized procedure will still put the value between 0 and 1. Arguably this approach might be more useful to distinguish relatively shallow networks from each other, but less useful when the networks predominantly focus on the object itself and individual differences are more related to the activation magnitude and not location. In line with this hypothesis is the observation that the differences between shallow and deep networks are mostly driven by less use of background features. Future studies could circumvent this bias by using methods that use the magnitude of the activations without normalizing the values. Another possible explanation is the omission of negative activations. Perhaps ResNet-34 used negative activations to suppress background features to a greater degree than ResNet-18.

Regardless of this inconsistency, we clearly observed that both ResNet-18 and 34 changed their strategy as a function of the contextual information and that especially ResNet-34 decreased the use of the background features relative to the object features when the scene was incongruent with the object.

2.6.7. Methods local maxima

Local maxima Grad-CAM

Finally, we investigated the number of local maxima in the activation maps. We used a maximum filter of 20 by 20 pixels with a threshold of the mean plus 1 standard deviation on the mask. The result was a black and white image with islands of local maxima. We then counted the number of local maxima per object by looking for curves that join continuous points (i.e. contours). In total, we used 1000 images per object category. We excluded the object category "speakers" since the category had multiple, separate, objects at a time (e.g., two speaker towers). For the comparison between networks, we reported a Bonferroni corrected t-test.

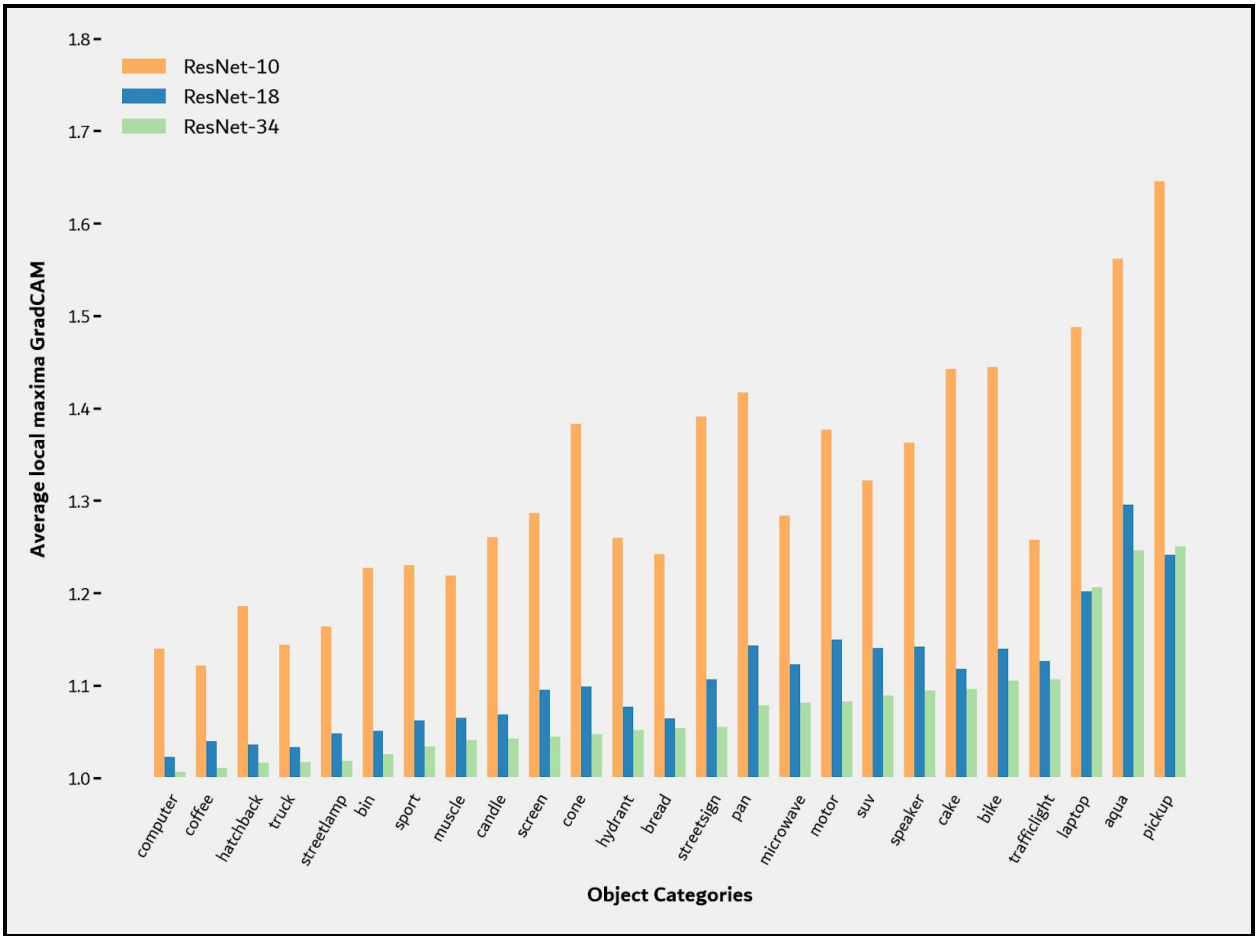


Figure 11: Average local maxima of the Grad-CAM activation map for ResNet-10, 18 and 34.

2.6.8. Results local maxima

Next, we counted the number of local maxima in the activation maps of each image. We interpreted the local maxima in the Grad-CAM activation map as a region of interest for the network. For

example, in Fig. 8e we can observe that ResNet-10 has multiple local maxima, whereas ResNet-18 has a single local maximum. We reasoned that in this example, ResNet-18 used all the object features, whereas ResNet-10 used individual object part features. Therefore, the analysis is a crude estimation if a network sees the object as a whole or simply a sum of parts. Here, we restricted ourselves to ResNet-10, 18 and 34 since the larger activation map of ResNet-6 inherently increased the number of local maxima (e.g. example 8e). The results are displayed in Fig. 11. Increased network depth was associated with fewer local maxima in the activation maps. We used a t-test to first compare ResNet-10 and 18 to see if the differences in the local maxima per category were different (two-sided t-test, corrected for multiple comparisons with Bonferroni). The t-test indicated that ResNet-18 had a significantly lower average local maxima for all categories. Then, we did the same for ResNet-18 and 34 and found that ResNet-34 had a significantly lower average local maxima in 12 out of 25 categories.

2.6.9. Discussion local maxima

We interpret the decrease in the number of local maxima as a shift in the complexity of the used features. Arguably, shallow networks rely more often on relatively lower-level features. Since the likelihood that these lower-level features appear in multiple spaces is higher, the shallow networks have on average more local maxima in their activation map. The deeper networks were more likely to use high-level feature constellations. These features were less likely to be present in multiple locations, thus on average, the deeper networks had less local maxima in their activation map. The difference in strategy can be illustrated through the example of a car, shallow networks might identify the car by detecting the wheels and the windows, whereas deeper networks look specifically for parts of the images that contain a wheel on the bottom, the car body in the middle, and the windows on top. The difference in the complexity of these features resulted in a difference in local maxima (two or more maxima vs. one local maximum).

2.6.10. General conclusion visualization and quantification Grad-CAM

Overall we can conclude that network depth allows networks to perform implicit scene segmentation through the selection of more object features relative to background features. Moreover, depth enables networks to see objects more as a whole instead of a sum of parts. We hypothesize that this ability is mediated by the ability to use the high-level features that consist of specific configurations of low-level features. To confirm that deep networks indeed use more high-level features we set out to remove the high-level features from the images while still retaining low-level features.

2.7. Experiment 3: The role of feature complexity in scene segmentation

2.7.1. Introduction

The Grad-CAM results suggested two structural differences between shallow and deep networks, first, we showed that, during classification, increased network depth resulted in the use of more object relative to background features. Second of all, the regions of interest in the classification process were more unified in space with increased network depth. The question then arises how DCNNs 'know' which features belong to the object. Based on our previous findings, we hypothesized that network depth enables networks to use high-level feature constellations for the classification processes and thereby effectively segmenting the object from the background. To test this hypothesis, we scrambled the images in our validation set. Scrambling the image will result in the disruption of high-level feature constellations, however, low-level features, such as textures and simple shapes, are still largely intact. Each image was divided into an 8 by 8 grid, subsequently, we scrambled the 64 grids to create a new image (see Fig. 12 for an example and the methods for more information). For humans, it is quite difficult to identify the object in such scrambled scenes. Determining if the image depicts an SUV or a hatchback is a tough exercise, since the features that separate the two are high-level (e.g. the relationship between the car's window, body and wheels and their corresponding sizes). For testing, we used the ResNets networks trained in experiment 2.

2.7.2. Methods

Scrambling method

Our goal was to remove the high-level features in the images while still retaining a large part of low-level features. To this end, we divided each image into a grid of equally large areas (e.g. width and height divided by 8). Then, we scrambled the parts randomly. As can be seen in Fig. 12 each block only contains a part of the object, and often the part is abruptly ended by the next block. High-level features that encompass the spatial relationship between individual parts are therefore unequivocally destroyed. To prevent that the scrambling process by chance recreates a part of the original image and thereby leaving high-level features intact we placed several restrictions on the scrambling features. In the scrambled image, parts could not be placed next to their original neighbour in the same order (horizontally and vertically). After scrambling the image, there were sharp lines throughout the image. To prevent that the networks interpret these artefacts as features we smoothed the transition by taking three pixels on each side of the transition and applied a Gaussian filter with a sigma of two standard deviations to the area. As a result, the transitions appeared much abrupt. We performed the scrambling experiment with 2x2 up until 12x12 blocks with intermittent steps of two. In the results we discuss the 8x8 scrambled condition, the rest of the scrambled conditions can be seen in the Supplementary Fig. 1 to 6.

Figure 12: Scrambled image example. The image depicts a hatchback car. Whereas the original image clearly displays a hatchback, the scrambled image is much harder to classify, especially considering the similarity of other categories (e.g. SUV).

2.7.3. Results and discussion

The results of the experiment are displayed in Fig. 13. Remarkably, ResNet-6 outperformed the deeper networks in the congruent condition. We thus see a complete reversal of the performance of previous experiments. Arguably, ResNet-6 (and ResNet-10 to a smaller extent) still managed to use the scene information to classify the objects correctly. All networks performed poorly on the scrambled

incongruent object-scene pairs (around near-chance). Scrambling the images resulted in a disruption of part of the features in the images, still, some scene features were virtually unaltered, for example, the scene type was not hard to infer when one looks at the textures (e.g. the texture of a road versus a wooden floor). For the incongruent condition, this meant that the wrong information was amplified and therefore even more disrupting. Near the 100% uninformative mark, performance for all networks collapsed. For these networks, the home and street scenes were never seen before.

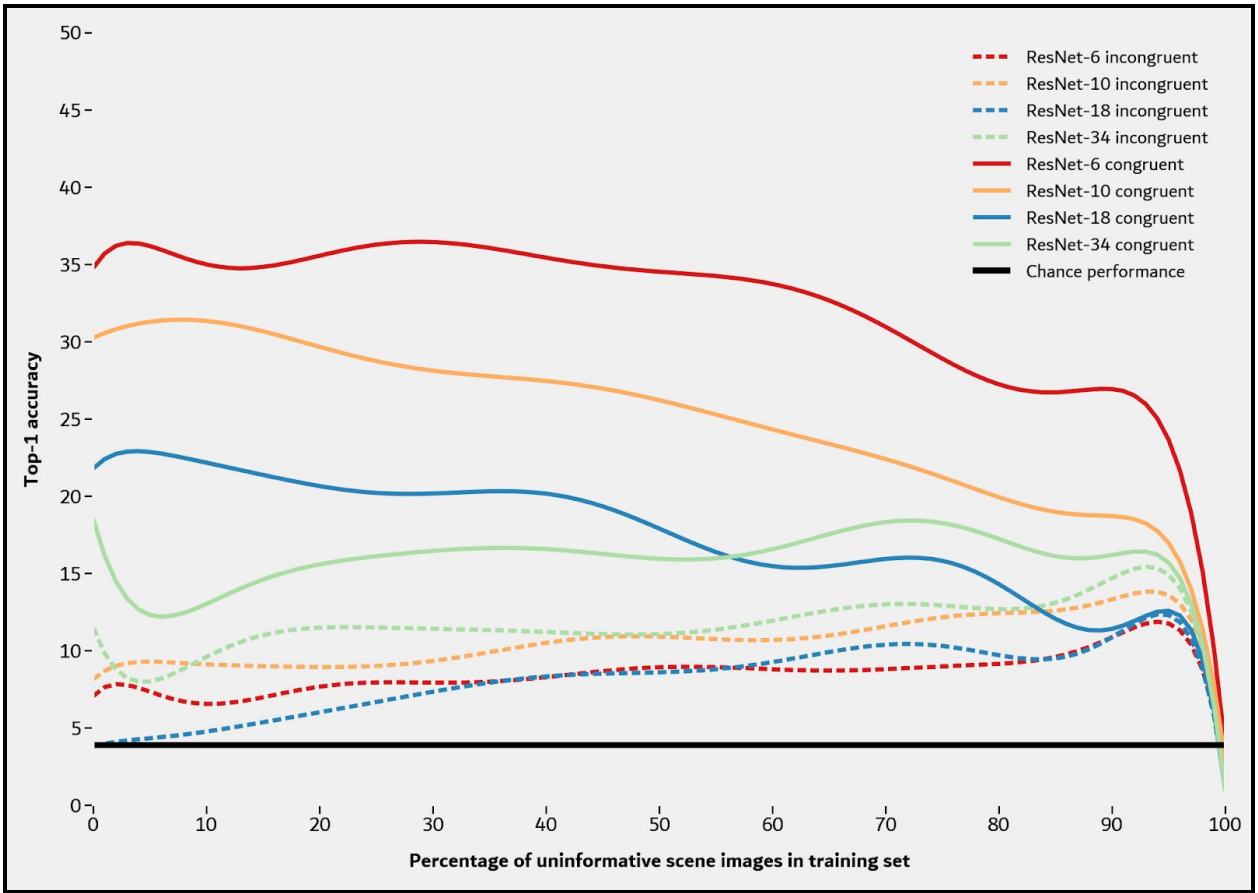


Figure 13. ResNet performance curve on scrambled images. The image was divided into a grid of 8 by 8 blocks and subsequently scrambled at random. The x-axis depicts the percentage of uninformative object-scene pairs in the training set. Note: the y-axis is from 0 to 50% top-1 accuracy so that the differences between the networks are better visible.

Apart from the reversal of the congruent condition performance, we see that the performance of the incongruent condition was slightly above chance for all networks. The poor performance is likely related to the inability to select high-level features that uniquely identify the target object. Whereas in

the congruent condition, the networks could use features in the background to gather evidence for a certain category, the evidence is now increasing the likelihood that lower-level object features are attributed to the wrong category. The background features are clearly still a deciding factor for identifying the object, especially when the network cannot accumulate sufficient evidence from the object features alone. The background features are disrupted to a smaller degree since the features are first of all generally smaller, and second of all less complex compared to object features (e.g. colors and textures are strong indicators of the scene type, these features are still clearly identifiable and it should be relatively easy for the networks to classify the scene). The poor incongruent condition performance thus shows that when necessary, all networks use background features. Under normal circumstances, deeper networks are influenced to a lesser degree by the background features when the networks can select features that distinguish one category from the others. Shallow networks are less capable of extracting the characteristic and thus rely more on background features.

The fact that we saw a complete reversal for the congruent condition to experiment 2 (i.e. performance is worse with increased depth) is crucial since we could attribute the difference in performance to the difference in activation map sizes. ResNet-6 has a higher spatial resolution (i.e. the neuron in the last layer looks at a smaller region of the image) and can by default not detect the entirety of features that encompass a large region. Therefore a part of the results could be explained by the fact that the scrambled image is relatively intact for ResNet-6 compared to the other networks. However, we see that ResNet-10 performed second to best in the congruent condition. ResNet-10 has an identical activation map size as ResNet-18 and 34, thus any differences that arise are due to the differences in network depth. These results were consistent over the scrambling condition 4x4 to 12x12 (see Supplementary Fig. 2 to 6). The only scrambling condition that deviated was the 2x2 condition (see Supplementary Fig. 1). In this condition, many high-level features were still present as the blocks contained ¼ of the image. Based on these results, we argue that deeper networks use more high-level features that encompass more space in the image. Since these features are inherently disrupted by our scrambling method, the performance has deteriorated to the degree that shallow networks that use lower-level features encompassing less space in the image outperform the deeper networks. These

results indicate that increased network depth enables networks to use high-level feature constellations since each additional convolutional operation increases the complexity of the features. These features are predominantly seen in the target object, therefore, selecting these features is effectively a way of segmenting the object from the rest of the scene.

2.7.4. Limitations

Even though our results clearly show that increased network depth increased the feature complexity, a few caveats have to be noted. First of all, the activation map of ResNet-6 is smaller than the activation map of ResNet-10, 18 and 34. This will by default limit the complexity of the extracted features. Luckily the rest of the ResNets share the same activation size and we found the same pattern between these networks. Secondly, our scrambling method is not the most elegant way to remove high-level feature constellations since it creates artificial boundaries between parts of the object. We tried to minimize this artefact by smoothing the edges, which indeed improved the performance slightly for all networks. Moreover, the scrambling method does not give us precise control over what features we remove as the way features are removed is random at the level of each individual feature[8]. Nonetheless, it does on average remove more high-level features than low-level features. Despite the somewhat crude approach, we believe that the exact reversal of the network performance is strong evidence for which type of features are used by the networks.

Nonetheless, it would be worthwhile to explore more elegant methods of scrambling the features in an image. One possible approach would be to scramble the object features within Unity. Depending on the object category and specific instance it might be possible to scramble different object parts throughout the scene (for example most 3d models of bikes are divided into different bike parts such as wheels and a steering frame). Other approaches might edit the properties of the images themselves so that certain features are removed, for example, Geirhos et al. (2019) created a stylized version of ImageNet to decrease the texture bias and increase the use of shape information.

[8] The process is random in the sense that the chance a feature is disrupted by the scrambling method is dependent on the exact location of the feature in the image. The images are randomly generated within the predefined parameters.

3. General Discussion

We investigated how the depth of DCNNs and contextual information affects object recognition performance. To this end, we systematically varied the contextual information. The first two experiments showed that increased network depth resulted in relatively large performance gains for the incongruent condition. Moreover, for the deep networks, the performance rapidly improved when the informativeness of the contextual information was decreased. We repeated the second experiment with the recurrent CORnet and showed that there was no increase in performance apart from what could be expected from the added depth when unrolling the recurrent network. Next, we visualized and quantified Grad-CAM saliency maps of the ResNets and showed that network depth increased the reliance on objects relative to background features. Deeper networks used larger areas of the object and the areas of interest were more united in space. To investigate the complexity of the features we selectively removed high-level features in the images by dividing the images into equal parts and scrambling those parts. Whereas deep networks outperformed shallow networks in both conditions, this result was now reversed for the incongruent condition as shallow networks outperformed deep networks.

Our results suggest that increased network depth enables DCNNs to select high-level object features and thereby implicitly segmenting the object from the background. Increased network depth allows the network to learn this ability even if the contextual information reliably predicts the object class during training. The mechanism enabling the scene segmentation is the selection of high-level features that are sufficiently complex so that they are only related to the object and never appear in the background. Furthermore, we found no concrete evidence that DCNNs with within-area recurrent connections improved the implicit scene segmentation beyond what can be expected from the increased network depth.

3.1. DCNNs provide an alternative to classical grouping and segmentation theories

While the DCNNs were never instructed to perform scene segmentation, we have provided compelling evidence for their ability to perform implicit scene segmentation and have shed light on its

underlying mechanism. Recently, DCNNs have been considered as models for the human visual system. Therefore, we could tentatively argue that our findings are in contrast to classical grouping and segmentation theories on how humans are theorized to perform such processes. These theories propose an explicit step where the object is segmented from the rest of the scene and subsequently grouped into one coherent object. For example, the feature-integration theory proposes a two-stage process (Treisman, 1999). In the first preattentive stage, basic features (e.g., color and shape) are processed automatically and in parallel. In the second attentive stage, other properties including relations between features of an object, are processed sequentially and ultimately bound together to create a single object that is perceived. The grouping and segmentation steps in the second stage are not present in DCNNs. Nevertheless, DCNNs are still able to perform scene segmentation similar to humans in outcome. Our results indicate that the explicit steps in the attentive stage might therefore—at least in theory—be unnecessary for core object recognition. Rather, the additional processing depth provided by recurrent connections might be sufficient for segmenting the object from the background.

3.2. CORnet is an unrolled feedforward network

We conducted experiment 2 with both feedforward ResNets and recurrent CORnets and did not find concrete evidence that simple within-area recurrent connections improved scene segmentation beyond the increased processing depth through recurrent processing. Kubilius et al. (2018) created CORnet to bring DCNNs anatomically closer to the brain. Kubilius et al. argue based on their own benchmark (Brain-Score) that CORnet is currently the best computational model for the brain. However, the benchmark appears to be highly biased towards CORnet as the benchmark includes a dataset that tested the temporal dynamics of neural data. CORnet achieved a correlation of .25 while the correlation of other feedforward networks was set to zero since the feedforward architecture cannot capture temporal dynamics. The Brain-Score is calculated by averaging over a total of three datasets. Unsurprisingly, CORnet beats out the other feedforward networks. However, based on the other tests CORnet does not perform better than other feedforward networks. We agree that ideally, the

architecture of a DCNN-based computational model includes recurrent connections. However, as a computational model, CORnet only shows that within-area recurrent connections are effectively a way of increasing depth of the network by reusing existing layers. Computationally speaking, the results of CORnet and feedforward architectures are identical and CORnet can therefore be seen as an unrolled feedforward network.

While CORnet appears to be a special case of a feedforward DCNN, other implementations of recurrent DCNNs might show different abilities. The CORnet implementation of recurrent processing lacks many crucial properties of recurrent processing in the brain. The CORnet architecture has within-area (local) but not between-area (global) recurrent connections, moreover, the architecture lacks lateral connections. To date, there are no DCNNs that implement recurrent processing in a biologically valid manner while coming close to the performance of feedforward networks on the ImageNet benchmark. However, for simple tasks, such as number recognition, recurrent networks can outperform feedforward networks under difficult circumstances such as occlusion (Spoerer, McClure, & Kriegeskorte, 2017; Kietzmann et al. 2019). The main bottleneck is the computational load, which makes it infeasible for the networks to train on high-resolution images. Taking this into consideration, we can not make any claims beyond our findings that simple feedforward processing of sufficient depth can result in a similar outcome as a network with simple within-area recurrent connections.

3.3. High-level features and shape

Our study showed that scene segmentation achieved by DCNNs occurs by the selection of high-level features. However, we have not looked at the properties of these features directly. There is still an ongoing debate on what type of features DCNNs can and cannot represent. When it comes to high-level features, shape is the most debated feature. Shape features can be classified as local and global shape. Local shape refers to the local contours of an object and can be seen as low-level features. The evidence for such local shape features is unquestionable, as a large part of the early layers of DCNNs are dedicated to shape perception (Olah et al., 2020a). Examples of such local shape features

can be found in Box 4. The image becomes murkier for global shape. Global shape is regarded as a high-level feature and thought to be an important cue for object recognition, particularly for living objects, for object recognition (Lloyd-Jones & Luckhurst, 2002; Elder & Velisavljević, 2009). Recent studies show that DCNNs cannot use abstract global shapes such as silhouettes (Baker, Lu, Erlikhman & Kellman, 2018 & 2020). The lack of this capability is directly related to the use of pooling operations and strided convolutions (Sabour, Frosst, N., & Hinton, 2017). If this holds, then what is the nature of the high-level features responsible for the implicit scene segmentation observed in this study?

For implicit scene segmentation to occur, the network needs to detect high-level feature constellations that are unique to the object. These high-level features are constructed out of lower-level features which are shared among other classes. The combinations of these lower-level features result in unique high-level feature constellations. These constellations encompass, among others, local shape features and spatial configuration of the features that make up the constellation. For example, see the circuit of neurons displayed in Fig. 3 of Box 4. The circuit is composed of car features such as wheels and windows that are subsequently combined into a neuron that looks for specific spatial configuration of these individual parts (Olah et al. 2020b). Such circuits could encode high-level features without the need for global shape.

While the lack of global shape processing will undoubtedly result in classification errors, it is still possible that the network performs implicit scene segmentation despite outputting the wrong category. This could occur when global shape plays an important role in distinguishing two highly similar classes from each other and the network consequently selects the wrong class. The features that the network selects are likely accumulating evidence for both classes and are located in the same location. The selection of these features, regardless of the final winner-takes-all answer, indicate which features belong to the object and background.

In sum, implicit scene segmentation in DCNNs can potentially occur despite its inability to process global shape.

3.4. Alternative recurrent network architectures

Up to this point, we have discussed DCNNs that all follow the core components of any convolutional architecture. While the feedforward architectures are extremely effective for its purpose, there are insurmountable limitations to the architecture as is, namely the apparent inability to process global shape. Geoffry Hinton, the godfather of deep learning and co-creator of AlexNet, the first successful DCNN, has been a vocal proponent of a different approach. Hinton argues that the pooling operation used in DCNNs is deeply flawed. As an alternative, Hinton and colleagues (Sabour, Frosst, N., & Hinton, 2017) proposed a new architecture, Capsule Neural Networks (CapsNet) to overcome the limitations of DCNNs. The architecture of CapsNet is based on the idea of cortical minicolumns in the brain. All neurons in such a column have the same receptive field and receive common input. In CapsNet each cortical minicolumn is a capsule that represents a single feature. The output is sent to a higher-level capsule that receives input from multiple capsules in the form of a prediction (whether or not a feature is present). After multiple iterations made possible by recurrent connections, the capsule converges to one solution and either indicates that a feature is present or not present (Hinton, Sabour & Frosst, 2018). CapsNet can capture the global shape of an object and are theorized to result in transformation equivariance instead of merely invariance. Whereas invariance captures properties that do not change as the result of a transformation, equivariance is the property that changes predictably. Equivariance gives the network the ability to extrapolate to previously unseen poses of an object. Recent studies indeed show that CapsNets are capable of global shape perception and can perform scene segmentation and grouping (Doerig, Schmittwilken, Manassi & Herzog, 2019 & 2020). While the idea and the results seem promising, CapsNet fails to perform on real-world datasets (Xi, Bing & Jin, 2017). CapsNet even falls short on CIFAR-10, a dataset with 10 classes and 32x32 resolution images, that is considered a trivial challenge for simple feedforward DCNNs. Training on more complex datasets is computationally too expensive. The studies of Doerig et al. were performed with simple stimuli and would not scale to more complex data. For this reason, we cannot use CapsNet for our data.

Doerig et al. (2020) note that CapsNet performs scene segmentation through the detection of simple features and the subsequent grouping of these features in higher-level features. The iterative routing by agreement process converges to one solution and therefore effectively segments the object from the background. This process is akin to what we observed in our study: scene segmentation in feedforward DCNNs where the network selects high-level features through a series of convolutional operations. The main difference is that CapsNets converge to a single solution and that DCNNs encode the probability that a certain feature is present in their receptive field. Both DCNNs and CapsNet thus perform scene segmentation through the selection of high-level features.

3.5. The flexibility of deep convolutional neural networks

Our current study showed that the choices concerning the training and validation set can heavily influence the performance of DCNNs. At best, an experiment can show what DCNNs are capable of—and not what they are not capable of. DCNNs have by definition a greedy approach—a network trained on a dataset with a lot of different textures will learn to classify objects based on textures. Seijdel et al. (2020) showed that with sufficient depth, networks can perform implicit scene segmentation. In our study, we showed that this ability was not always present in the networks, but rather arose when the amount of contextual information decreased. Networks of sufficient depth increasingly relied on object features instead of background features and thereby managed to increase the performance rapidly when the amount of contextual information decreased. By creating performance curves we were able to show how networks adapt to different circumstances. The changes in strategy emphasize an important point, namely that DCNNs are versatile and their solutions are closely related to the training data. By systematically altering the training data, we were able to reveal the adaptiveness of the networks in a novel way.

4. General limitations

We have discussed the limitations regarding the first two experiments in the results section for the specific experiments. Here, we discuss the limitations of the last experiment as well as possible solutions for future studies.

4.1. Attribution techniques

Grad-CAM is an intuitive visualization method which helps to attribute which parts of the images contributed to the final decision. The downside to the method is that the spatial resolution of this approach is limited by the size of the activation map of the final convolutional layer. In practice, this meant that the regions of interest were often larger than the objects themselves (for examples see Fig 9). For this reason, we could only roughly determine which pixels were attributed to the decision. To circumvent this limitation, we identified the pixels belonging to the object and subsequently determined which pixels were in the direct proximity of the object. These neighbouring pixels were subsequently disregarded in our analysis. This approach got rid of false positives in the data, however, it inadvertently lowered the number of pixels we could use in our analysis. We do not think this is a major issue in terms of the interpretability of the results, as all networks (10, 18 and 34) were equally impacted and the exclusion arguably omits a part of the relevant data, thus making it harder to find differences. Despite these limitations, we did find clear differences between the networks.

For this study, we opted for Grad-CAM due to its widespread use within the field of computer vision. To increase the explainability of DCNNs, there is an active research field investigating which attribution techniques and implementations provide activation maps with the highest resolution with as little noise and bias as possible. For example, the recently released Semantic Input Sampling for Explanation (SISE) promises to improve the previous state-of-the-art on all these metrics (Sattarzadeh et al., 2020). For future research, we advise the adoption of multiple state-of-the-art techniques to avoid that one method might bias the results. Ideally, the used methods are diverse in their approach as each

technique might highlight a different aspect of the classification process (e.g. gradient, guided backprop and perturbation based methods).

While attribution is certainly an important part of the toolbox, the approach has weaknesses when used in isolation. The biggest drawback of only using saliency maps is that pixels are not per se meaningful and are entangled with neighbouring pixels (Olah et al., 2018). Moreover, the amount of information in the saliency maps is limited as only one class can be shown at a time. These weaknesses can be improved by combining attribution with feature visualization techniques. By doing so, we can investigate the network with greater detail. Instead of seeing which pixels contributed to the classification, we can now see what features the network detects at that location. An example of such an interface can be seen in Supplementary Fig. 7. The only downside to this approach is that the interfaces that combine attribution with feature visualization techniques are scarcely available and all need considerable tweaking to meet our intended purposes.

5. Future directions

Based on our findings there are multiple interesting directions for future research. In general, the results argue that it is worthwhile to see which capabilities that are generally attributed to recurrent processing can be performed by feedforward DCNNs. This will in turn point to the areas where DCNNs are lacking. Next, researchers can adjust the architecture (e.g. adding a different type of recurrent connections) to investigate which anatomical details enable the ability.

In the next paragraphs, the future directions directly related to our study are discussed.

5.1. High-level object feature properties

In our study, we linked the ability to perform scene segmentation in DCNNs to the selection of high-level features. As discussed in the general limitations of our current study we think that combining the use of attribution and feature visualization techniques is a promising research direction. These techniques in particular make DCNN such a powerful scientific model as we can explore the inner

mechanisms to an unprecedented degree. By doing so, we use DCNNs in the same capacity as animal models are used to understand the human brain (Scholte, 2018). The unique possibilities of DCNNs could allow us to understand which type of features are used for implicit scene segmentation.

In the present study, we have not looked at how the network selected features when the classification was successful vs. unsuccessful. In future studies, it would be interesting to explore why DCNNs fail to classify an object correctly and how this relates to the selection of high-level features.

5.2. Background feature properties

In this study, the networks were heavily influenced by the appearance of the background scene, suggesting that the chosen scene types were fundamentally different from each other. This raises the question: what properties of the scene do DCNNs use to identify the scene? Do DCNNs use other objects to identify the scene—or rather for example simple summary statistics? In this regard, DCNNs could potentially fundamentally differ from humans. Whereas humans only see a part of their visual field in focus, DCNNs see every part of the image with the same resolution. Future studies could investigate what properties of the background are used by DCNNs and how this relates to human visual processing. Moreover, it would be interesting to control for the input differences.

Another interesting research direction is how the background changes the strategy itself. Are DCNNs simply processing the scene and subsequently converging the evidence, or do the networks weigh the evidence differently as a function of certain cues? From human psychophysical experiments, we know that visual processing is influenced by low-level contrast statistics of the scene. These statistics signal the need for further processing through recurrent connections (Groen et al., 2018). Could feedforward networks do something similar computationally? Or can only recurrent networks approximate these solutions?

6. Conclusion

In the here presented studies we show that increased depth of DCNNs allows for the selection of high-level feature constellations unique to the object. Through this mechanism, DCNNs implicitly segment the object for the rest of the scene. Recurrent connections improve the implicit scene segmentation via additional processing depth. Given these findings, we argue that the mechanisms of implicit scene segmentation in DCNNs might be similar to scene segmentation through recurrent processing in humans.

7. Miscellaneous methods

Packages

For data processing and analysis, we made use of freely available python libraries, including *cv2*, *NumPy*, *Pandas*, *scikit-image*, *SciPy*, *researchpy*, *statsmodels*, *PIL* and *PyTorch*. For the Grad-CAM experiment, we adapted this GitHub repository by WonKwang Lee. Data processing and analysis were done with the help of *Jupyter Notebooks*. Visualizations were created with *Matplotlib*, *Adobe Photoshop* and *Google Presentations*.

Network training

Experiments 1, 2 and 3 were conducted on the Lisa cluster computer of the University of Amsterdam with multiple NVIDIA 1080 Ti GPUs. The Grad-CAM experiments were conducted on Paperspace with a single NVIDIA Quadro P5000.

Networking training dependencies

The networks were trained with Python version 3.6.6-intel-2019b, CUDA version 10.0.130 and cuDNN version 7.4.2-CUDA-10.0.130.

Training scripts availability

The Python training and testing scripts are available here. Note, the scripts were executed through SLURM batch jobs.

Data availability

The data and the corresponding Juypter notebooks to process and analyze the data are available here.

References

Azulay, A., & Weiss, Y. (2018). Why do deep convolutional networks generalize so poorly to small image transformations?. arXiv preprint arXiv:1805.12177.

Baker, N., Lu, H., Erlikhman, G., & Kellman, P. J. (2018). Deep convolutional networks do not classify based on global object shape. *PLoS computational biology*, *14*(12), e1006613.

Baker, N., Lu, H., Erlikhman, G., & Kellman, P. J. (2020). Local features and global shape information in object classification by deep convolutional neural networks. *Vision Research*, *172*, 46-61.

Cadena, S. A., Denfield, G. H., Walker, E. Y., Gatys, L. A., Tolias, A. S., Bethge, M., & Ecker, A. S. (2019). Deep convolutional models improve predictions of macaque V1 responses to natural images. *PLoS computational biology*, *15*(4), e1006897.

Cichy, R. M., & Kaiser, D. (2019). Deep neural networks as scientific models. *Trends in cognitive sciences*, *23*(4), 305-317.

Davenport, J. L., & Potter, M. C. (2004). Scene consistency in object and background perception. *Psychological science*, *15*(8), 559-564.

Doerig, A., Schmittwilken, L., Sayim, B., Manassi, M., & Herzog, M. H. (2020). Capsule networks as recurrent models of grouping and segmentation. *PLoS computational biology*, *16*(7), e1008017.

Elder, J. H., & Velisavljević, L. (2009). Cue dynamics underlying rapid detection of animals in natural scenes. *Journal of Vision, 9*(7), 7-7.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., & Brendel, W. (2018). ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*.

Groen, I. I., Jahfari, S., Seijdel, N., Ghebreab, S., Lamme, V. A., & Scholte, H. S. (2018). Scene complexity modulates degree of feedback activity during object detection in natural scenes. *PLoS computational biology, 14*(12), e1006690.

He, K., Zhang, X., Ren, S., & Sun, J. (2016, October). Identity mappings in deep residual networks. In *European conference on computer vision* (pp. 630-645). Springer, Cham.

Hinton, G. E., Sabour, S., & Frosst, N. (2018, February). Matrix capsules with EM routing. *In International conference on learning representations*.

Hochstein, S., & Ahissar, M. (2002). View from the top: Hierarchies and reverse hierarchies in the visual system. *Neuron*, *36*(5), 791-804.

Howe, P. D. (2017). Natural scenes can be identified as rapidly as individual features. *Attention, Perception, & Psychophysics*, *79*(6), 1674-1681.

Kietzmann, T. C., McClure, P., & Kriegeskorte, N. (2019). Deep neural networks in computational neuroscience. In *Oxford research encyclopedia of neuroscience*.

Kietzmann, T. C., Spoerer, C. J., Sörensen, L. K., Cichy, R. M., Hauk, O., & Kriegeskorte, N. (2019). Recurrence is required to capture the representational dynamics of the human visual system. Proceedings of the National Academy of Sciences, 116(43), 21854-21863.

Kindel, W. F., Christensen, E. D., & Zylberberg, J. (2019). Using deep learning to probe the neural code for images in primary visual cortex. *Journal of vision*, *19*(4), 29-29.

Kriegeskorte, N. (2015). Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual review of vision science*, *1*, 417-446.

Kriegeskorte, N., & Douglas, P. K. (2018). Cognitive computational neuroscience. *Nature neuroscience*, *21*(9), 1148-1160.

Kubilius, J., Schrimpf, M., Nayebi, A., Bear, D., Yamins, D. L., & DiCarlo, J. J. (2018). Cornet: Modeling the neural mechanisms of core object recognition. *BioRxiv*, 408385.

Lamme, V. A., & Roelfsema, P. R. (2000). The distinct modes of vision offered by feedforward and recurrent processing. *Trends in neurosciences*, *23*(11), 571-579.

Laskar, M. N. U., Giraldo, L. G. S., & Schwartz, O. (2020). Deep neural networks capture texture sensitivity in V2. *Journal of vision*, *20*(7), 21-1.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

Liao, Q., & Poggio, T. (2016). Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*.

Lloyd-Jones, T. J., & Luckhurst, L. (2002). Outline shape is a mediator of object recognition that is particularly important for living things. *Memory & Cognition, 30*(4), 489-498.

Neisser, U., & Becklen, R. (1975). Selective looking: Attending to visually specified events. *Cognitive psychology, 7*(4), 480-494.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., & Carter, S. (2020a). An overview of early vision in inceptionv1. Distill, 5(4), e00024-002.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., & Carter, S. (2020b). Zoom In: An Introduction to Circuits. Distill, 5(3), e00024-001.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., & Mordvintsev, A. (2018). The building blocks of interpretability. *Distill*, *3*(3), e10.

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*, *2*(11), 1019-1025.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Berg, A. C. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211-252.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in neural information processing systems* (pp. 3856-3866).

Sattarzadeh, S., Sudhakar, M., Lem, A., Mehryar, S., Plataniotis, K. N., Jang, J., ... & Bae, K. (2020). Explaining Convolutional Neural Networks through Attribution-Based Input Sampling and Block-Wise Feature Aggregation. *arXiv preprint arXiv:2010.00672*.

Scholte, H. S., Jolij, J., Fahrenfort, J. J., & Lamme, V. A. (2008). Feedforward and recurrent processing in scene segmentation: electroencephalography and functional magnetic resonance imaging. *Journal of cognitive neuroscience*, *20*(11), 2097-2109.

Scholte, S. (2018). Fantastic DNimals and where to find them. *Neuroimage, 180*, 112-113.

Seijdel, N., Tsakmakidis, N., De Haan, E. H., Bohte, S. M., & Scholte, H. S. (2020). Depth in convolutional neural networks solves scene segmentation. *PLoS computational biology*, *16*(7), e1008022.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (pp. 618-626).

Serre, T., Oliva, A., & Poggio, T. (2007). A feedforward architecture accounts for rapid categorization. *Proceedings of the national academy of sciences*, *104*(15), 6424-6429.

Spoerer, C. J., McClure, P., & Kriegeskorte, N. (2017). Recurrent convolutional neural networks: a better model of biological object recognition. *Frontiers in psychology*, *8*, 1551.

Treisman, A. (1999). Solutions to the binding problem: progress through controversy and convergence. *Neuron*, *24*(1), 105-125.

Van Essen, D. C., & Maunsell, J. H. (1983). Hierarchical organization and functional streams in the visual cortex. *Trends in neurosciences*, *6*, 370-375.

VanRullen, R., & Thorpe, S. J. (2001). Is it a bird? Is it a plane? Ultra-rapid visual categorisation of natural and artifactual objects. *Perception*, *30*(6), 655-668.

Wyatte, D., Curran, T., & O'Reilly, R. (2012). The limits of feedforward vision: Recurrent processing promotes robust object recognition when objects are degraded. *Journal of Cognitive Neuroscience*, *24*(11), 2248-2261.

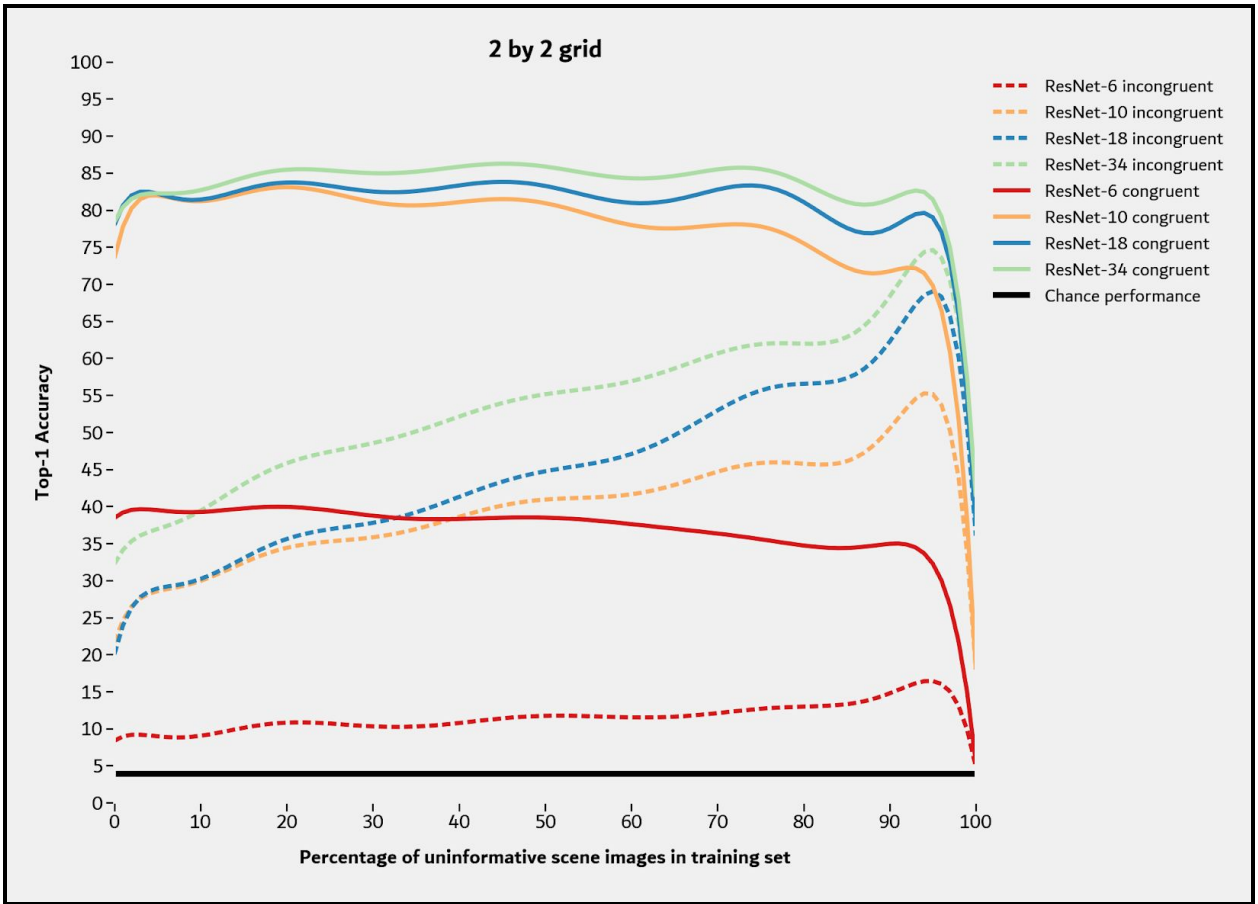
Xi, E., Bing, S., & Jin, Y. (2017). Capsule network performance on complex data. *arXiv preprint arXiv:1712.03480*.

Yamins, D. L., & DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, *19*(3), 356-365.

Yamins, D. L., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, *111*(23), 8619-8624.

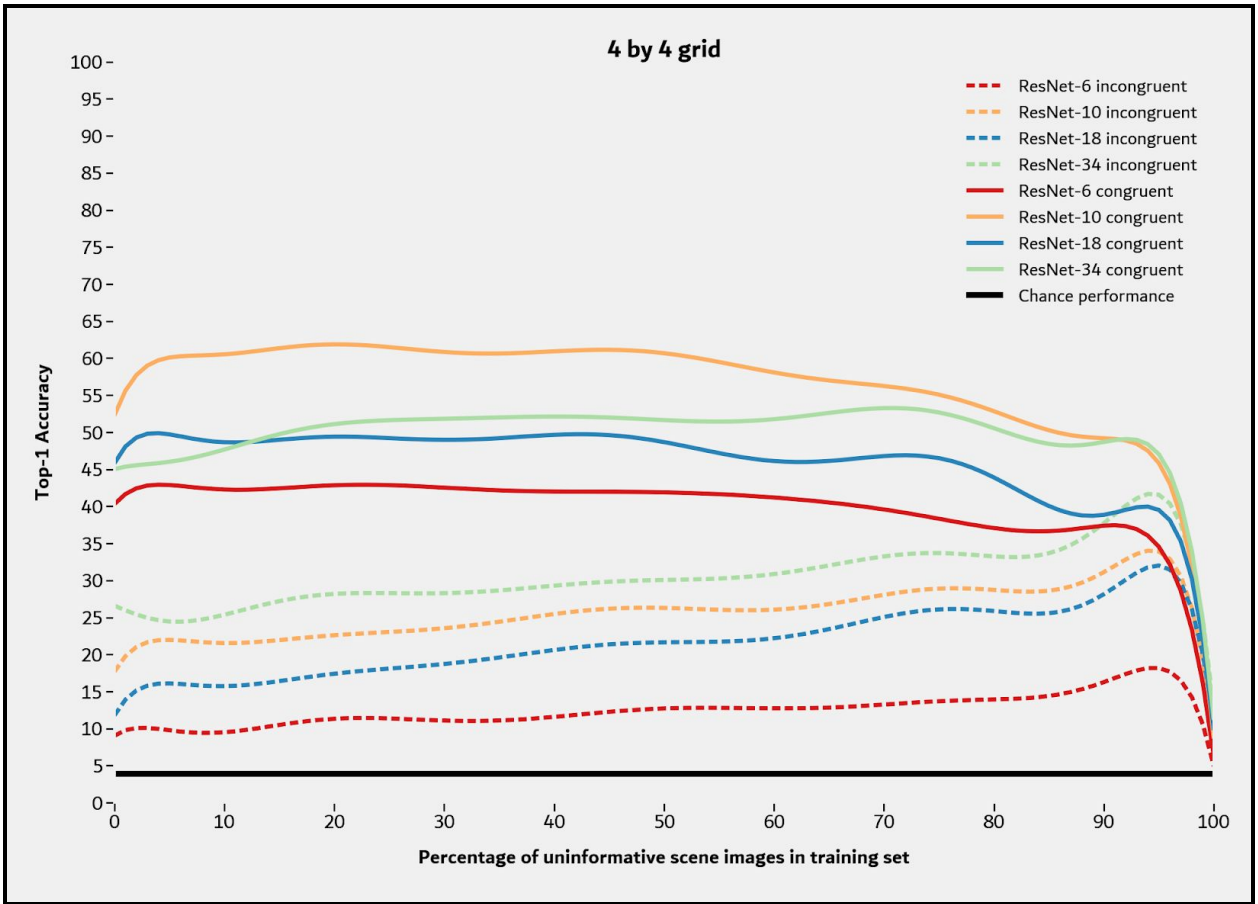
Zeman, A. A., Ritchie, J. B., Bracci, S., & de Beeck, H. O. (2020). orthogonal Representations of object Shape and category in Deep convolutional neural networks and Human Visual cortex. *Scientific Reports*, *10*(1), 1-12.

Supplementary Figures



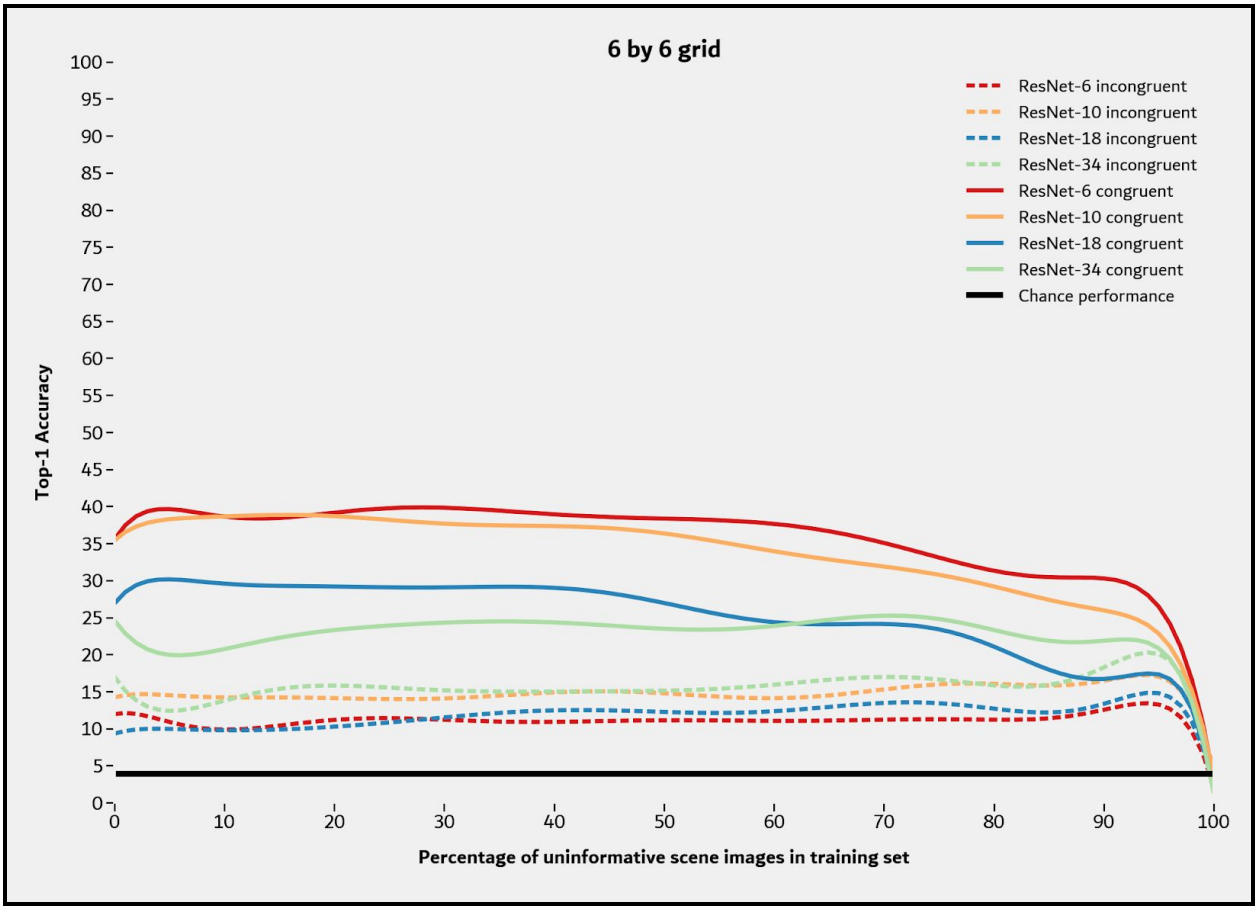
Supplementary Figure 1: ResNet performance curve on scrambled images (2 by 2 grid). The image was divided into a grid of 8 by 8 blocks, the blocks were subsequently scrambled at random. The x-axis depicts the percentage of uninformative object-scene pairs in the training set.

Back to results



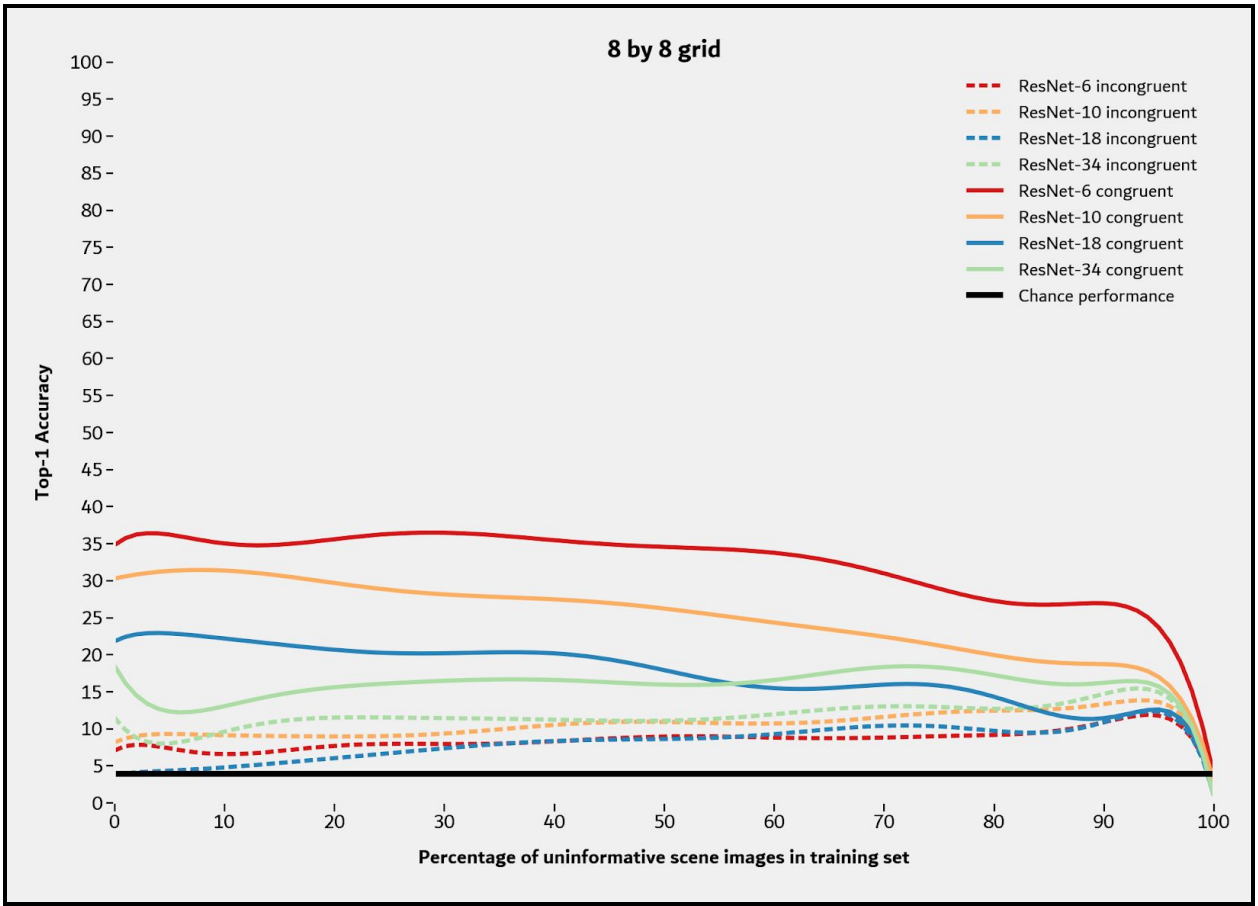
Supplementary Figure 2: ResNet performance curve on scrambled images (4 by 4 grid). The image was divided into a grid of 4 by 4 blocks, the blocks were subsequently scrambled at random. The x-axis depicts the percentage of uninformative object-scene pairs in the training set.

Back to results



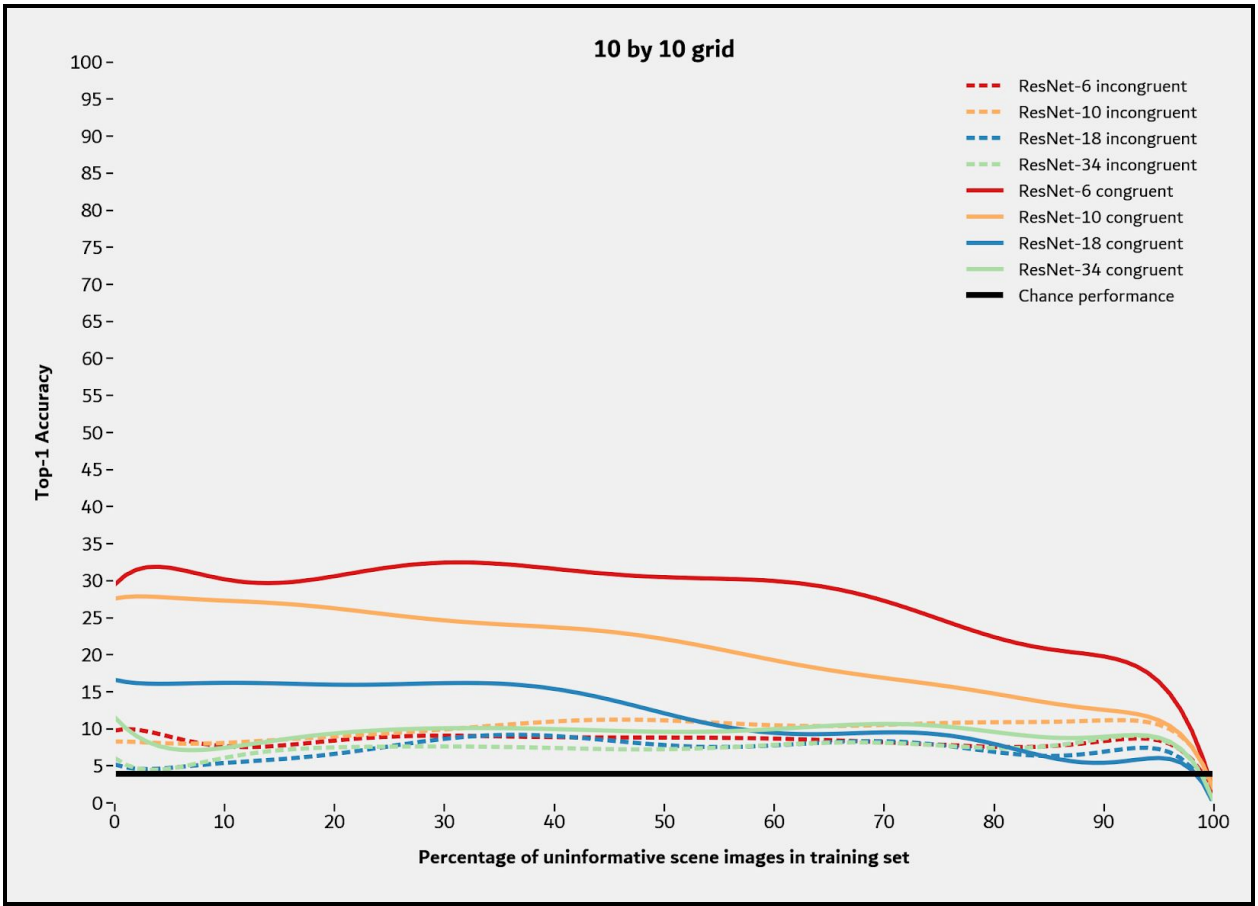
Supplementary Figure 3: ResNet performance curve on scrambled images (6 by 6 grid). The image was divided into a grid of 6 by 6 blocks, the blocks were subsequently scrambled at random. The x-axis depicts the percentage of uninformative object-scene pairs in the training set.

Back to results



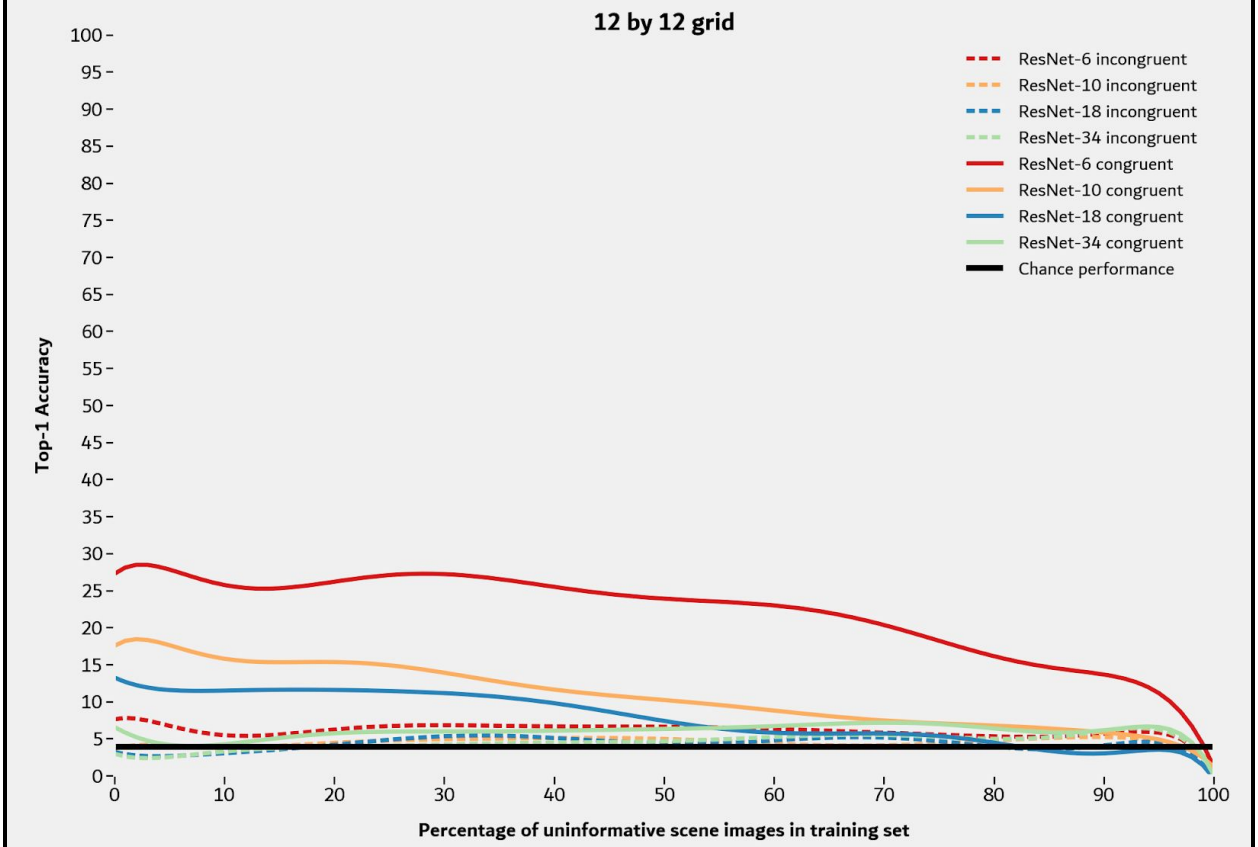
Supplementary Figure 4: ResNet performance curve on scrambled images (8 by 8 grid). The image was divided into a grid of 8 by 8 blocks, the blocks were subsequently scrambled at random. The x-axis depicts the percentage of uninformative object-scene pairs in the training set.

Back to results



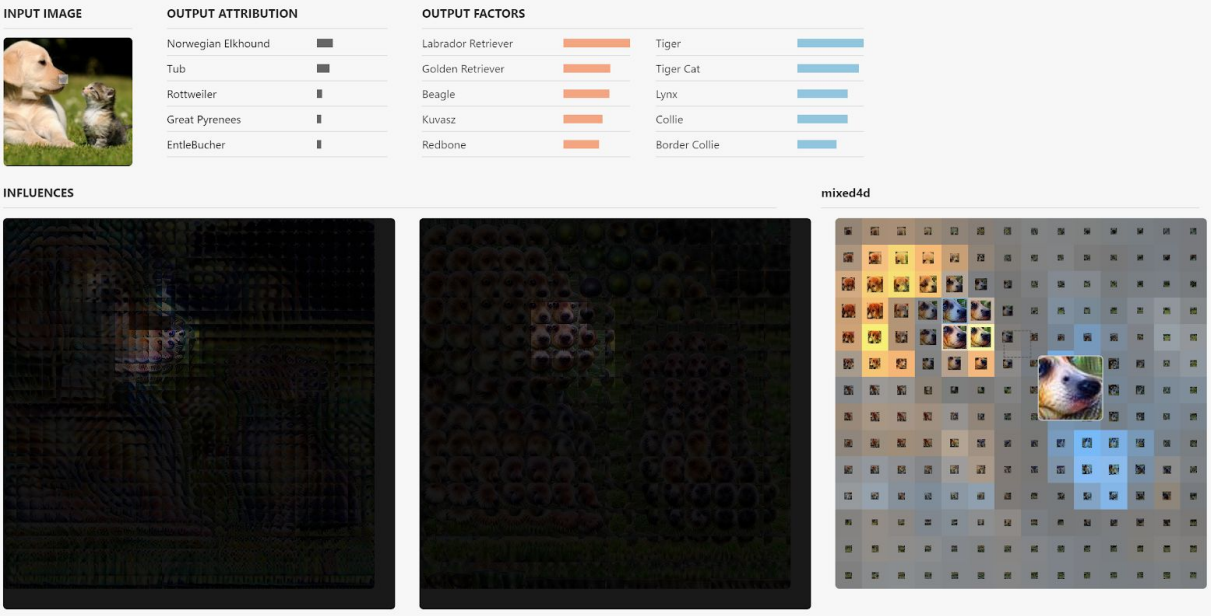
Supplementary Figure 5: ResNet performance curve on scrambled images (10 by 10 grid). The image was divided into a grid of 10 by 10 blocks, the blocks were subsequently scrambled at random. The x-axis depicts the percentage of uninformative object-scene pairs in the training set.

Back to results



Supplementary Figure 6: ResNet performance curve on scrambled images (12 by 12 grid). The image was divided into a grid of 12 by 12 blocks, the blocks were subsequently scrambled at random. The x-axis depicts the percentage of uninformative object-scene pairs in the training set.

Back to results



Supplementary Figure 7: Example of an interface combining feature visualization with attribution techniques. In the figure, we see how attribution techniques can be combined with feature visualization techniques into one interface. The approach leads to a greater understanding of the network than the sum of its parts. In this example, we investigate what the network sees when looking at a region of the image (proportional to the activation map size of the layer). In this example, the snout of the dog is selected (top left of the interface). The activation map on the right shows what the network at layer *mixed4d* makes of each region of the activation map. The selected region is displayed on top of the grid. Each image in the grid is sized proportionally to the magnitude of the activations. In this example, we can clearly see that the network detects a dog snout in the selected region. Moreover, the feature visualization shows the rough shape of the snout and the spatial relation to other features (e.g. eyes and mouth). The activation maps in the middle and on the left display which part of the image (i.e. neurons) influence the decision. Finally, it is interesting to note that we see that the network selects a high-level feature that is unique to the target object (i.e. it is unlikely to appear in the background) yet shared among different object classes (e.g. labrador/golden retriever). Olah et al. (2018) under Creative Commons Attribution CC-BY 4.0.

Back to discussion