

Seminar Visual Computing

Learning Deep Object Detectors From 3D Models

Julian Karlbauer*
Universität Ulm

ABSTRACT

Labeling data for neural network training is a time consuming procedure. In regard of creating huge amounts of training data for novel object categories, Peng et al.'s [8] work takes part in the idea of using free available 3D CAD models to synthetically generate training data. Keeping in mind that CAD models tend to lack in detail e.g. missing texture or unrealistic pose, [8] evaluates to which extent DCNNs are invariant to those details. In further investigation, surprising results legitimize Peng et al.'s approach to efficiently generate training data for new object categories. They found out, that CNNs develop invariances against specific low-level cues of image data and thus can be used to augment existing training data to achieve even better performance in object detection.

1 INTRODUCTION

Computational object detection takes a state of fundamental significance in computer science. Some AI applications for example, that interact with real world environments demand a working object detection mechanism for further computation. Deep CNN models "achieve state of the arts performance on object detection"[8] and are therefore often trained to do so. Unfortunately training data needs to be labeled e.g. each object has to be manually marked with a bounding box which is highly time consuming. Already labeled training data does exist but is limited in categories. Examples of such datasets are provided by: PASCAL VOC [2], COCO [6] or ImageNet[1] wherein ImageNet, with 200 categories, happens to provide by far the largest dataset. Introducing new labeled categories therefore can be necessary and is attended with a lot of effort. The creators of [8] face that specific problem presenting new ideas stated in the following. This work is based on [8]'s authors ideas and will summarize and discuss their concepts.

They put the idea to heavily reduce time consumption caused by collection and annotation by automatically generating training data using freely available 3D CAD models. Websites like 3dwarehouse.sketchup.com grant free access to huge amounts of 3D modeled objects that can be used to train Deep CNN networks for object detection. That would not be the first approach of using synthesized data to train object detectors. [10] already trained a small amount of categories but has not demonstrated how this approach combines with many different and new categories, which is a main objective in [8]. However, they furthermore state that the main challenge, following this approach, is the fact that freely available CAD models may capture the correct 3D shape of an object but often do lack specific so called 'low-level cues' such as (realistic)texture, background, realistic pose etc. This, they say, can cause serious issues using a DCNN because of its ability to retain those low-level cues in its internal object representation. Other works[11] used HOG (Histogram Of Gradients) based architectures to detect textureless objects, which worked well due to the fact that HOGs are fully invariant to object texture but mainly focus on edge detection. Still,

it is stated, that HOGs in general are less powerful than CNNs – resulting in a overall lower detection rate. Keeping this issue in mind [8] explores "how missing low-level cues affect DCNNs ability to learn object detection". Therefore they define 'cue invariance' as "the ability of the network to extract the equivalent high-level category information despite missing low-level cue"[8]. Accordingly, the question arises if CNNs can possibly be invariant to missing low-level cues and if yes, to which degree? An approach to that question will eventually follow up and leads to interesting new domains including the use of synthetic data in CNN training and object detection.

To acquire further insight, the authors developed a method to efficiently generate huge amounts of useful training data from 3D CAD models. Using the synthesized training data leads to surprising discoveries, as the authors state. They realize that DCNNs "encode far more complex invariances to cues such as 3D pose, color, texture and context than previously accounted for"[8]. Further, they quantify, how learned invariances correlate with the specific training task. It can even be shown that this approach outperforms common training procedures like [11] when available training data is limited or missing for new object categories. Along the way the authors gain important insights in DCNN behavior, show that DCNN detection performance can be increased by their method and present a large-scale evaluation of synthetic CAD training of object detectors.

2 FURTHER ARRANGEMENTS

Convolutional neural networks (CNNs) consist of several layers, usually some convolutional layers at the beginning, each of which forms a special filter to preprocess and simplify the input data, followed by few fully connected layers that mainly perform classification or data generation tasks [4]. To make a CNN develop appropriate filters and to achieve reasonable outputs on data the network has never seen before, it is trained on a carefully selected dataset that ideally covers all characteristics of the function that shall be approximated [9]. Consequently, during training phase, a CNN – just as any other artificial neural network – is taught to repeat correct outputs and change incorrect outputs in such a way to provide more suitable answers in future.

[8] explains that object detection long time was handled with HOG or SIFT alike algorithms due to they can be considered invariant to factors like translations, illumination, contrast and other low-level cues as they localize the global outlines of an object and base their detection on that. Anyhow, upcoming DCNNs overtake these procedures in many categories, including object detection. Not at least because of their complex encoding of high-, mid-, and low-level cues. The authors refer that "DCNNs learn layered features starting with familiar pooled edges in the first layer, and progressing to more and more complex patterns with increasing spatial support"[8]. CNNs can also be extended with inclusions like 'sliding window CNN' or 'Regions-CNN' (RCNN) which, by using detection algorithms, splits up given images into regions of interest and then feeds them into a CNN to make it detect objects even more accurate. Accuracy can be increased by up to 30 percent with this method[3].

Despite investments in CNN research, we still have problems in

*e-mail:julian.karlbauer@uni-ulm.de

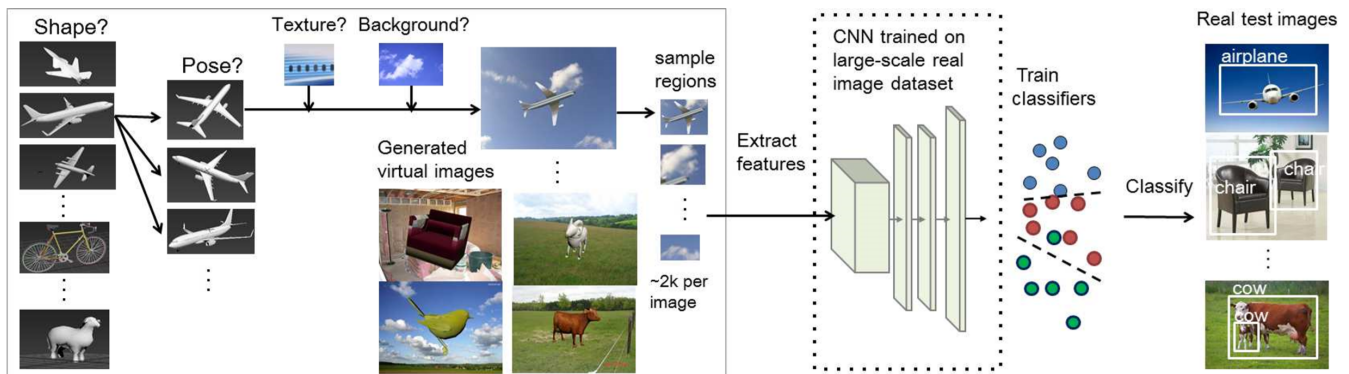


Figure 1: An overview of peng2015learning et al.'s approach to synthetically generate training data and train a CNN [8]

understanding DCNNs functionality and why they even work as well as they do. The creators of [8] highlight that there especially "has been increasing interest in understanding the information encoded by the highly nonlinear deep layers." and bring in related work that are concerned in investigate these features. [14] and [13] which made further investigation and bring some clearance into this specific area can be named here. Specific functionality of isolated parts of a CNN are part these works. Particularly interesting for the referred paper is the work of [7] being commented with "Their visualizations confirm that a progressively more invariant and abstract representation of the image is formed by successive layers". Still they do not spotlight the nature of the invariances. However, this shows that complex representation of invariances in DCNNs is a matter of fact as assumed earlier. This will further play an important role when it comes to the use of 3D CAD models, which can be noisy and low quality. The use of such data is nothing new to computer vision. As the authors state, quite a view papers exist determine the value of usability of e.g. CAD models in training object detectors. However, [8] goes deeper and the authors want to "evaluate our approach on all 20 categories in the PASCAL VOC2007 dataset, which is much larger and more realistic than previous benchmarks", which no one has ever done before, making the paper the largest-scale evaluation in using CAD models for object detection. Later on, related papers will be discussed in regard to how to classify Peng et al.'s work.

3 APPROACH

Peng et al.'s main goal now has crystallized to first – find a method to synthetically generate huge amount of training data in a short amount of time, consuming little human effort and secondly – develop an approach to be able to evaluate CNNs invariance against low-level cues. With knowing to what degree CNNs are invariant to those cues, it would be easier to understand if CNNs can use non photo realistic data to learn detecting real objects.

As already suggested, to generate data, freely available 3D CAD models from 3dwarehouse.sketchup.com will be used. An overview of the approach is shown in Fig.1. Throughout a few-step process, data is generated as explained below. After 2D images are generated, the training set gets fed into a CNN. The data generated by the CNN will then be used to train the final object classifiers.

3.1 Data Generation

Peng et al. explain that "realistic object appearance depends on many low-level cues, including object shape, pose, surface color, reflectance, location and spectral distributions of illumination sources, properties of the background scene, camera characteristics, and others". Online available 3D CAD models often lack many of those features. For example, if you find a 3D model of

a dog, it most likely is projected in an unnatural pose. It rather looks like a stiff doll than a real dog. Additionally, many 3D models do not come with texture, they are rather in a uniform grey. Peng et al. chose to use simple computer graphic mechanisms to generate low level cues such as texture, background, rotation, pose – and more. To that task Peng et al. mention "when learning a detection model for a new category with limited labeled real data, the choice of whether or not to simulate these cues in the synthetic data depends on the invariance of the representation. For example, if the representation is invariant to color, grayscale images can be rendered", meaning that if CNNs turn out to be invariant to low-level cues, ultimately no low-level cues need to be generated.

CAD Models In the first step, 3D CAD models are downloaded. To train a specific category e.g. 'Dog', several dog models are downloaded. To have a pool of different initial models, about 5-25 models are downloaded for each category to train. Synthetically adding cues to these models will then generate a vast amount of different training models.

Positioning After collecting the data, viewpoints are generated. 3D CAD models easily can be positioned and rotated in every direction by using 3D modeling programs. To optimally train the CNN, realistic postures need to be generated. A chair tilted upside down would not make too much sense. Therefore a few standard views, that fit the categories nature, are manually adjusted. Normally about 3 to 4 standard views are created i.e. front-view, side-view and a few intra-views. To not make it too simple, they add small perturbations, meaning a small randomized value to again rotate the object in a certain direction.

Texture Third step is to add realistic texture to the models. To achieve that, several images, containing real texture, are downloaded. In case of the dog example, a few real dog images are collected and boxes are drawn around texture areas. Those areas get extracted and later put on the texture-less CAD models.

Background 2D rendering images would now lead to a grey-ish background, which could interfere the learning process of the neural net. We still do not know how invariant the CNN is to image background. Due to that, suitable background images are collected. e.g. for the dog example – meadow, living room, street-like pictures are collected and put behind the CAD models, so that, if eventually rendered, a natural background is generated. For each category around 40 background images are collected to obtain a realistic setup.

Generate 2D Image With the models being put up to a usable

training data pool, 2D images can be generated automatically. To achieve that, random combinations of the previous steps are computationally set up and then 2D rendered to at least a 1000 images per training category.

3.2 Determine Invariances

[8] says that HOG based object detectors previously have shown invariances against object background and texture. However, they "hypothesise that the case is different for DCNN representations, where neurons have been shown to respond to detailed textures, colors and mid-level patterns, and explore the invariance of DCNNs to such factors". They further explain that for HOG based detectors tests have been made, where objects had uniform-grey textures or backgrounds. Peng et al. want to proceed similar and state to use several training sets, each containing different low-level cues. Background wise they want to use three different types. First, a real RGB image imitating a natural low-level cue, second, a greyscale image, testing the CNN against background color invariance and third, a plain white background, determining its invariance against a background at all. To determine texture invariances of the object itself they use two types of textures. On the one hand, a real RGB texture to, again, imitate a real low-level cue and on the other hand a uniform grey texture to check how the CNN reacts to missing low-level cues texture wise.

Invariance was defined as the ability to name correct high-level cues (e.g. category) despite missing low-level cues (e.g. texture etc.) of an specific object. Transferred to the function of a CNN that would mean that, with missing low-level cues, specific low-level neurons of the net would not be activated. Still, if the CNN is invariant to that cue, similar or the same high-level neurons should react as if the low-level cues were existent. To precise that, Peng et al. plan to use two training sets for each invariance to be studied. One training set will contain the specific low-level cues and the other will not. For example, if the invariance to object texture is investigated, a training set with real RGB texture and a training set with uniform grey texture will be generated. A CNN will be trained on both data sets and the results are compared against each other. Peng et al. say that if the neural net is invariant to, let's say, cat texture, in both cases the neural net will show equal high-level neuron activation. That would mean, that the net will "[...]hallucinate' the right texture when given a textureless cat shape". Detectors will then "learn cats equally well from both sets of training data" they state. On the other hand, if the CNN is not invariant to the specific cue, the cue-less version will eventually perform worse when put to an object detector.

3.3 AlexNet

To execute their Experiments, Peng et al. use the well known AlexNet that has been designed by Alex Krizhevsky. AlexNet is a widely used visual convolutional network. It consists of 8 overall layers of which the first five are convolutional layers and the last three layers are fully connected to each neuron of the previous layer. This leads to a network with over 60 million parameters. "This network had first achieved breakthrough results on the ILSVRC-2012 image classification, and remains the most studied and widely used visual convnet" the authors accentuate. It further "is trained by fully supervised backpropagation [...] and takes raw RGB image pixels of a fixed size of 224 x 224 and outputs object category label"[8]. This network also was used for the development of the RCNN [3] that the authors often refer to.

Expectations [8]'s authors expect the net to learn "different cue invariances depending on the task and categories it is trained on". They assume that the net will choose different focuses, depending on unique properties of the training data. They name the example of a leopard, which has a unique texture and therefore the

net will rather focus in that cue than on outer shape. Its invariance against texture will then almost be non-existent. To see how task specific training affects the net, they use three different versions of it. "1) one pre-trained on the generic ImageNet ILSVRC 1000-way classification task (IMGNET); 2) the same network additionally fine-tuned on the PASCAL 20-category detection task (PASC-FT); and 3) for the case when a category has no or few labels, we fine-tune the IMGNET network on synthetic CAD data (VCNN)"[8]. The VCNN network is obtained by fine-tuning it on synthetic data by backpropagating the gradients with a lower learning rate, because "this has the effect of adapting the hidden layer parameters to the synthetic data"[8] and it allows the network to learn more specific information about object categories, so it can develop invariances. Now it can be investigated, how CNNs can learn from synthetic data with the presence of different low-level cues.

4 RESULTS


Peng et al. executed several experiments to have the CNN react to different cue combinations. Their first evaluation is about cue invariances while the second is about few-shot scenarios.

4.1 Cue Invariances

Peng et al. investigate a few different types of cues. First – background, color and texture. Second – 3D object pose, by using different rotations. Later they will also investigate real image pose by using only one view category, and 3D shape by reducing the amount of CAD models used during training.

Background, color and texture For a setup the authors chose to use all view positions while using 1-2 perturbations per view. As mentioned above, a vast combination of object textures and image backgrounds was generated and used for the execution. To make the experiment reliable, they set up some specific configurations. Namely they put up RR-RR, W-RR, W-UG, RR-UG, RG-UG, RG-RR with RR = real RGB, W = white, UG = uniform grey, RG = real grey. The first element tells us the background color and the second element stands for the used object texture. The procedure put together would be to "1) select cues, 2) generate a batch of synthetic 2D images with those cues, 3) sample positive and negative patches for each class, 4) extract hidden DCNN layer activations from the patches as features, 5) train a classifier for each object category, 6) test the classifiers on real PASCAL images and report mean Average Precision (mAP)"[8]. Training the net with RR-RR images and watch its mAP results in a peek performance at around 2000 training examples.

It comes out that training with synthetic data generally results in a less precise (less mAP) object detector compared to training with real image data, which was expected the authors state. Also, the IMGNET network competes worse than the PASCAL network as it was already stated in previous works [3]. Anyhow, Peng et al. are surprised "that the generation settings RR-RR, W-RR, W-UG, RG-RR with PASC-FT all achieve comparable performance". Further, setups with no background color and real texture compete best (RG-RR, R-RR), resulting in the cognition that the network "has learned to be invariant to the color and texture of the object and its background"[8]. On the other hand the setups RR-UG, RG-UG perform worst. Possibly because "the uniform object texture is not well distinguished from the non-white backgrounds" they state. The IMGNET network delivers a similar trend, but the best performing setups here are RR-RR, RG-RR, meaning that the IMGNET classifier profits from real textures and at the same time the network is less invariant to these cues. A further interesting point is that RG-RR achieves good results in each of the networks which "leads to the conclusion that both networks have learned to associate the right context colors with objects". As already expected, results can differ, depending on the training



	RR-RR	W-RR	W-UG	RR-UG	RG-UG	RG-RR
BG	Real RGB	White	White	Real RGB	Real Gray	Real Gray
TX	Real RGB	Real RGB	Unif. Gray	Unif. Gray	Unif. Gray	Real RGB

PASC-FT	aero	bike	bird	boat	botl	bus	car	cat	chr	cow	tab	dog	hse	mbik	pers	plt	shp	sofa	trn	tv	mAP
RR-RR	50.9	57.5	28.3	20.3	17.8	50.1	37.7	26.1	11.5	27.1	2.4	25.3	40.2	52.2	14.3	11.9	40.4	16.3	15.2	32.2	28.9
W-RR	46.5	55.8	28.6	21.7	21.3	50.6	46.6	28.9	14.9	38.1	0.7	27.3	42.5	53.0	17.4	22.8	30.4	16.4	16.7	43.5	31.2
W-UG	54.4	49.6	31.5	24.8	27.0	42.3	62.9	6.6	21.2	34.6	0.3	18.2	35.4	51.3	33.9	15.0	8.3	33.9	2.6	49.0	30.1
RR-UG	55.2	57.8	24.8	17.1	11.5	29.9	39.3	16.9	9.9	35.1	4.7	30.1	37.5	53.1	18.1	9.5	12.4	18.2	2.1	21.1	25.2
RG-UG	49.8	56.9	20.9	15.6	10.8	25.6	42.1	14.7	4.1	32.4	9.3	20.4	28.0	51.2	14.7	10.3	12.6	14.2	9.5	28.0	23.6
RG-RR	46.5	55.8	28.6	21.7	21.3	50.6	46.6	28.9	14.9	38.1	0.7	27.3	42.5	53.0	17.4	22.8	30.4	16.4	16.7	43.5	31.2

IMGNET	aero	bike	bird	boat	botl	bus	car	cat	chr	cow	tab	dog	hse	mbik	pers	plt	shp	sofa	trn	tv	mAP
RR-RR	34.3	34.6	19.9	17.1	10.8	30.0	33.0	18.4	9.7	13.7	1.4	17.6	17.7	34.7	13.9	11.8	15.2	12.7	6.3	26.0	18.9
W-RR	35.9	23.3	16.9	15.0	11.8	24.9	35.2	20.9	11.2	15.5	0.1	15.9	15.6	28.7	13.4	8.9	3.7	10.3	0.6	28.8	16.8
W-UG	38.6	32.5	18.7	14.1	9.7	21.2	36.0	9.9	11.3	13.6	0.9	15.7	15.5	32.3	15.9	9.9	9.7	19.9	0.1	17.4	17.1
RR-UG	26.4	36.3	9.5	9.6	9.4	5.8	24.9	0.4	1.2	12.8	4.7	14.4	9.2	28.8	11.7	9.6	0.7	4.9	0.1	12.2	11.6
RG-UG	32.7	34.5	20.2	14.6	9.4	7.5	30.1	12.1	2.3	14.6	9.3	15.2	11.2	30.2	12.3	11.4	2.2	9.9	0.5	13.1	14.7
RG-RR	26.4	38.2	21.0	15.4	12.1	26.7	34.5	18.0	8.8	16.4	0.4	17.0	20.9	32.1	11.0	14.7	18.4	14.8	6.7	32.0	19.3

Figure 2: Detection results on the PASCAL VOC2007 test dataset. Each row is trained on different background and texture configuration of virtual data shown in the top table. In the middle table, the DCNN is trained on ImageNet ILSVRC 1K classification data and finetuned on the PASCAL training data; in the bottom table, the network is not fine-tuned on PASCAL. [8]

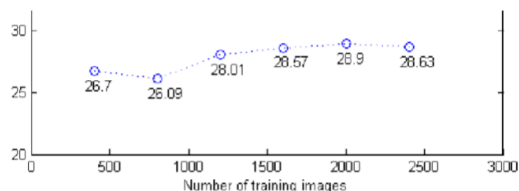


Figure 3: Peek performance occurs at around 2000 training images [8]

task. 'Cat' or 'Sheep' examples for instance, highly depend on texture and background cues, as it is shown for 'Sheep' in Fig. 4. As the numbers in the top left corner of each image tell, detection rate lowers, when texture and/or background gets removed. In these examples the network potentially learned to focus on specific texture or background details and therefore is less invariant these cues.

Thus, CNN react differently to specific training tasks. Still, it can be observed, that CNNs do have the potential to develop invariances against certain cues. On the other hand it is hard to understand to which extent CNNs encode low-cue information.

3D pose The authors of [8] call out, that it already is known, that CNNs develop invariances against small translations and scaling. However, it is not known how they react to complex 3D rotation. Peng et al.'s approach is to generate dominant rotations (e.g. front-, side-, intraview). The amount of each category changes with repeating training, keeping the overall training images the same amount. Additionally small perturbations (-15 to 15 degree) are added in. It results that adding front and side view improves the detectors accuracy, while adding an intraview only does little to no impact. They furthermore note that "adding some views may even hurt performance (e.g., TV) as the PASCAL test set may not have objects in those views". [8]

Real image pose Here the question is, to which extent data can be left out, so that the detector still reaches comparable results. Being more specific, Peng et al. want to fully remove e.g. side-view images from the training data and then perform equally on front-view and side-view detection. To investigate the idea, they put together training data as mentioned above, but with the distinction that only images with a considerable front-, sideview difference are used. Images of round objects (e.g. bottle, ball etc.) do not make it in the training data, while images of cars for example do. The surprising result is, that the detector in each case almost reaches the same mAP as it previously did.

3D shape The last investigation Peng et al. made, is to reduce 3D CAD model input and observe the result. It came out that, when lowering the CAD model usage by 50 percent, mAP lowers by 6.4 percent. This, the authors say, "shows a significant boost from adding more shape variation to the training data, indicating less invariance to this factor". [8]

These results are indeed interesting. To summarize their observation they state that they "found that DCNNs learn a significant amount of invariance to texture, color and pose, and less invariance to 3D shape, if trained (or fine-tuned) on the same task. If not trained on the task, the degree of invariance is lower. Therefore, when learning a detection model for a new category with no or limited labeled real data available, it is advantageous to simulate these factors in the synthetic data." [8]

4.2 Zero-Shot Learning Results

In this section the authors made up a final investigation. Few-Shot learning means to train a network with only a small amount of real labeled image data. In this evaluation Peng et al. simulated a zero-shot situation with literally no real image data, resulting in a pure 3D CAD model training. This basically is the most interesting part of Peng et al.'s work, because it finally shows how synthetically generated data can be used to completely replace labeling effort of real image data. To make the impact of the zero-shot scenario visible, they decided to also put up training examples with very little



Figure 4: Overlay of the unit's receptive field is drawn in white and normalized activation value is shown in the upper-left corner. For each unit we show results on (top to bottom): real PASCAL images, RR-RR, W-RR, W-UG. See text for further explanation. [8]

real image data. They effectively generate data pools with 5, 10 or 20 real images contained in it.

Results It finally results that, with limited real image pools, Peng et al.'s method perform better than the previously introduced RCNN. Also HOG features get outperformed. For the experiments RR-RR and RG-RR combinations had been used. The experiment had shown that RR-RR setups perform significantly worse than RG-RR. Potentially because of easier object separation on a grey background. Having in mind, that this method reaches almost as good performance as training on the PASCAL training set while only having a fraction of the effort to create it "speaks to the power of transferring deep representations and suggests that synthetic CAD data is a promising way to avoid tedious annotation for novel categories"[8]

5 CONCLUSION ON PENG ET AL.'S WORK

Peng et al. have investigated to which degree synthetic generated images can be used to train object detectors. As a source of synthetic image generation they used freely available 3D CAD models from the web. Due to lacking quality of many models, they put up a concept to efficiently transform them into usable training data, using simple computer graphic mechanisms. A leading question throughout the investigation was, to which degree missing low-level cues of synthetically generated training data will affect a CNN learning process. And following to that will lead to a less powerful object detector. Experiments have then shown that CNN indeed are able to develop invariances against specific low-level cues, still its degree is always dependent on the training task itself. Further experiments have shown, that Peng et al.'s method obtains potential when it comes to situations where labeled training data is limited. When augmenting small training data with synthetically generated data their method leads to remarkable good results. Thought further, this method can lead to lowering effort to create training data and therefore might be of particular importance in future applications.

6 RELATED WORK AND CLASSIFICATION

Peng et al. mention it themselves, "the use of synthetic data has a longstanding history in computer vision". Their idea to take 3D CAD images to help train CNNs also is a prominent concept. As mentioned, several works have already displayed an approach to this idea. [5] [10] [12] already tried the same, using a limited pool of 3D CAD models like cars, bicycles and motorcycles. Peng et al.'s work on the other hand gets weight, because they try to generalize the concept for any object category making a leap towards future training models where many different object classes might play a more important rule. The mentioned works above, as opposed to Peng et al., brought in the concept of internally 3D-represent an object after detecting it on a 2D image, which might be a powerful

tool in future AR applications i.e. in traffic, where partly hidden cars could be internally reconstructed by the program and thus resulting in a better overview. However, Peng et al. shows which impact 3D-positioning of objects has on CNN training, keeping the 3D-aspect of the used CAD models in mind very well. That leads towards the use of optimal training data and following to that to more efficient CNN training.

Accordingly, [8]'s work is based on already existing works that show the use of CAD models in CNN training for object detection. Anyhow, Peng et. al. augment the idea, bring in new important concepts and present a large scale evaluation, which makes their work a relevant part of this specific field of research. Based on [8]'s results, future thoughts can build up powerful tools to make use of synthetic data.

7 THOUGHTS AND FUTURE

The principal idea presented by Peng et al. seems plausible from the very beginning. Their explanations are based on fundamental knowledge and research, which makes their work interesting and good to read. The authors themselves don't actually put in further leading thoughts on their work and keep reserved towards presenting concepts or possibilities for the future. Still, they mention that their concept will be useful in further investigation.

Large scale training models might play an important role in the future, regarding to upcoming object detection models like self driving cars and other intelligent systems related to human computer interaction. But also computer graphic related programs will presumably be dependent on large-scale trained CNNs. Peng et al. may not present a solution to those large-scaled concepts but at least an approach to face the upcoming challenges. An approach means to be further examined to develop into a practical concept, which indeed puts a base for future work in this field of research. Presumed that their idea will have been fully evolved at some point would result in powerful training processes and thus mighty applications with the capability of recognizing every standard object we use in our daily activities. To be more precise; to accomplish such systems, there is no known way around synthetic data generation, which makes Peng et. al.'s work a leading presentation of future based thoughts.

ACKNOWLEDGEMENTS

This work is fundamentally based on Peng et al.'s work Learning Deep Object Detectors from 3D Models [8]

REFERENCES

- [1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.
- [2] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [3] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [5] J. Liebelt and C. Schmid. Multi-view object class detection with a 3d geometric model. In *CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition*, pages 1688–1695. IEEE Computer Society, 2010.
- [6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [7] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.
- [8] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep object detectors from 3d models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1278–1286, 2015.
- [9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- [10] M. Stark, M. Goesele, and B. Schiele. Back to the future: Learning shape models from 3d cad data. In *Bmvc*, volume 2, page 5. Citeseer, 2010.
- [11] B. Sun and K. Saenko. From virtual to reality: Fast adaptation of virtual object detectors to real domains. In *BMVC*, volume 1, page 3, 2014.
- [12] M. Sun, H. Su, S. Savarese, and L. Fei-Fei. A multi-view probabilistic model for 3d object classes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1247–1254. IEEE, 2009.
- [13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.
- [14] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.