Object Detector Differences When using Synthetic and Real Training Data

Martin Georg Ljungqvist[1][a], Otto Nordander[1], Arvid Mildner[2], Tony Liu[2] and Pierre Nugues[2][b]

[1]*Axis Communications AB, Lund, Sweden*

[2]*Department of Computer Science, Lund University, Lund, Sweden*

Keywords: Object Detection, Layer Similarity, Centered Kernel Alignment.

Abstract: To train well-behaved generalizing neural networks, sufficiently large and diverse datasets are needed. Collecting data while adhering to privacy legislation becomes increasingly difficult and annotating these large datasets is both a resource-heavy and time-consuming task. An approach to overcome these difficulties is to use synthetic data since it is inherently scalable and can be automatically annotated. However, how training on synthetic data affects the layers of a neural network is still unclear. In this paper, we train the YOLOv3 object detector on real and synthetic images from city environments. We perform a similarity analysis using Centered Kernel Alignment (CKA) to explore the effects of training on synthetic data on a layer-wise basis. The analysis captures the architecture of the detector while showing both different and similar patterns between different models. With this similarity analysis we want to give insights on how training synthetic data affects each layer and to give a better understanding of the inner workings of complex neural networks. The results show that the largest similarity between a detector trained on real data and a detector trained on synthetic data was in the early layers, and the largest difference was in the head part.

1 INTRODUCTION

Using convolutional neural networks (CNNs) is a popular approach to solve the object detection problem in computer vision. A lot of effort has been put into developing accurate and fast object detectors leveraging the structure of convolutional layers (Liu et al., 2016; Lin et al., 2017; Redmon and Farhadi, 2018; Tan et al., 2020). This has led to a drastic increase in performance of object detectors during the past few years. However, these models generally require massive amounts of labeled training data to achieve good performance and generalization (Nowruzi et al., 2019). Building these datasets can be both time consuming and resource heavy.

First, the raw data needs to be collected, often involving complex data acquisition setups and gathering schemes. Adhering to privacy, data protection regulations and ensuring the diversity and quantity of the data becomes an increasingly difficult challenge.

Second, the data needs to be annotated. Since datasets for deep learning often include several thousand images, the annotation process becomes a very mundane, time-consuming, and error-prone task.

One way of avoiding these issues is using synthetic data for training. Generated synthetic datasets are inherently scalable and labelling of the data can be done automatically. These datasets can for example be generated using a data generation tool such as Carla (Dosovitskiy et al., 2017), or sampling videos from open-world video games like *Grand Theft Auto V (GTA V)* (Richter et al., 2017; Johnson-Roberson et al., 2017).

A general problem with deep neural networks is that their complexity makes it difficult to understand exactly why a certain prediction has been made. This has led to neural networks often being considered as *black boxes* (Alain and Bengio, 2016; Fong and Vedaldi, 2017), where one only looks at the input and the output, while relying on trial and error when creating a well-working system. CNNs are less regarded as black boxes since they are suitable for visualisation, but that renders a vast amount of information to overview and may not tell everything about the networks inner workings. There have been many studies on understanding and visualizing the inner workings of deep neural networks (Hardoon et al., 2004; Zeiler and Fergus, 2014; Alain and Bengio, 2016; Fong and Vedaldi, 2017; Raghu et al., 2017; Morcos et al., 2018; Kornblith et al., 2019; Zhang et al., 2019;

[a] https://orcid.org/0000-0002-0115-869X

[b] https://orcid.org/0000-0002-9563-4000

Ljungqvist, M., Nordander, O., Mildner, A., Liu, T. and Nugues, P.
Object Detector Differences When using Synthetic and Real Training Data.
DOI: 10.5220/0010778200003124
In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 4: VISAPP*, pages 48-59
ISBN: 978-989-758-555-5; ISSN: 2184-4321

Hermann and Lampinen, 2020; Nguyen et al., 2021; Ge et al., 2021).

In this work, we investigate how object detection models are affected when trained on synthetic data versus real data by exposing the inner workings of the network. One key element will be the comparison between the outputs from individual hidden layers in the models using the recently proposed idea of similarity measurement (Kornblith et al., 2019). Our work builds upon Liu and Mildner (2020).

Our aim is to investigate how synthetic data affects the performance of object detection models as well as how hidden layers in the CNNs are affected by different types of training data. More specifically:

1. How does a model trained on synthetic data differ from one trained on real data and what network layers are affected?
2. Does freezing the backbone affect this?

To the best of our knowledge, no such analysis has been made on a CNN object detector using real and synthetic data.

Our main contributions are:

- We show what parts of the network are most similar for a detector trained on real image data compared to when it is trained on synthetic data.
- We also determine the consequences of freezing the backbone or not when further training a detector on synthetic data.

2 RELATED WORK

2.1 Object Detection

One-stage detectors are suitable for use in real-time object detection in video. These methods sample densely on the set of object locations, scales, and aspect ratios. Proposed methods are for example YOLO (Redmon and Farhadi, 2018), RetinaNet (Lin et al., 2017), SSD (Liu et al., 2016) and EfficientDet (Tan et al., 2020). These networks are significantly faster while having comparable performance to the conventional two-stage methods. Because of its speed, comparable accuracy, and relatively light-weightness, YOLOv3 (Redmon and Farhadi, 2018) was chosen for our experiments.

2.2 Synthetic Data

There are several synthetic datasets of city environments available and several experiments of training on synthetic data have been conducted. VKITTI (Gaidon et al., 2016; Cabon et al., 2020) is a synthetic version of the KITTI dataset (Geiger et al., 2013), but it does not contain persons. Synthia (Ros et al., 2016) is another synthetic dataset of images from urban scenes, where the results showed increased performance when training on a mixture of real and synthesized images. The video game GTA V has been used to generate synthetic datasets (Richter et al., 2017; Johnson-Roberson et al., 2017).

The experiments conducted in Johnson-Roberson et al. (2017) showed that training a Faster R-CNN on a GTAV synthetic dataset of at least 50,000 images increased the performance compared to training on the smaller real dataset Cityscapes (Cordts et al., 2016) when evaluated on the real KITTI dataset (Geiger et al., 2013). However, these experiments only used cars as labels, disregarding other labels such as persons and bicycles.

The Synscapes dataset is a synthetic version of Cityscapes (Wrenninge and Unger, 2018). The authors claim that training on only Synscapes yields decent results, but lowers performance compared to training on real data when evaluated on Cityscapes. However, their experiments showed that models trained on Synscapes outperformed both models trained only on GTAV (Richter et al., 2017) and Synthia (Ros et al., 2016).

Furthermore, Wrenninge and Unger (2018) claimed that training on a mixture of synthetic and real data can further improve performance, outperforming models trained only on real data. Results from Nowruzi et al. (2019) showed that training on synthetic data and fine-tuning on real data yielded better performance than training on a mixed real-synthetic dataset. The authors also concluded that photo-realism in the synthetic data was not necessarily as important as other factors in the training such as diversity.

Non-artistically generated images have been produces by *domain randomization* (Tremblay et al., 2018), where parameters such as lighting, pose, and textures were randomized. The authors showed that with additional fine-tuning on real data, their model outperformed models trained only on real data for object detection of cars on the KITTI dataset. Furthermore, they argued that letting the backbone be trainable during training on synthetic data yielded better performance compared to freezing the backbone weights.

Synthetic data have been used for pedestrian detection and pose estimation (Hattori et al., 2018). The authors showed that training on synthetic images only yielded a model that outperformed a model trained on real data only. However, the models were scene-specific and location-specific where they used a priori knowledge about the camera parameters and the scene geometry.

Hinterstoisser et al. (2018) superimposed 3D rendered models of toys with different lighting and poses onto real backgrounds. As opposed to Tremblay et al. (2018), the authors argued that freezing backbone weights (when they are initialized from a pretrained backbone) during training on the synthetic data yielded better performance compared to letting the backbone be trainable. The authors of Tremblay et al. (2018) argued that a possible explanation could be that the dataset that they used was large and diverse enough to further improve the backbone.

2.3 Similarity of Neural Networks

One way of obtaining more insight on how a CNN network behaves is looking at the outputs layer-wise. By comparing layer outputs from two different models, one can determine the similarity between the layers. One method of measuring the similarity of layer outputs is the *singular value canonical correlation analysis* (SVCCA) (Raghu et al., 2017). SVCCA uses *singular value decomposition* (SVD) (Golub and Reinsch, 1971) for dimensionality reduction and then *canonical correlation analysis* (CCA) (Hardoon et al., 2004) which was previously used to learn semantic representations for web images. A further improvement of SVCCA is the *projection weighted CCA* (PWCCA) (Morcos et al., 2018), which uses projection weighting to calculate the similarity measure as a weighted mean instead of a naive mean as in SVCCA.

Both metrics are invariant to invertible linear transformations which according to Kornblith et al. (2019) leads to some major issues. Kornblith et al. (2019) instead proposed a metric called *centered kernel alignment* (CKA) which, according to the authors, better captures similarity representations between network layers.

Later work (Raghu et al., 2017; Morcos et al., 2018) have shown that the Euclidean distance is not an ideal measurement of similarity between hidden layer outputs, but it can still give some useful insights.

While there exist several papers that attempt to answer how initialization, model complexity, or dataset size affect the similarity between models (Raghu et al., 2017; Morcos et al., 2018; Kornblith et al., 2019), no attempts have been made to compare the difference between models trained on synthetic and real data. As CKA gives a layer-wise similarity of hidden layers within the network, it can give insights of how such networks differ from each other on a layer-basis. These insights could be leveraged for example during training to target specific layers inside networks to improve performance.

3 MATERIALS AND METHODS

3.1 Datasets

3.1.1 Berkeley Deep Drive

The Berkeley Deep Drive (BDD) dataset (Yu et al., 2020) consists of 100,000 driving images collected from 50,000 rides, with 720p resolution. The images were collected from diverse scenes such as cities, residential areas, and highways, recorded during different hours of the day and in different weather conditions. The images are annotated with bounding boxes and class label.

20,000 out of the 100,000 images are reserved for the test set. Since the labels for the test set are unavailable, we use only the remaining 80,000 images for our experiments randomly divided into 60/20/20% for training, validation, and testing[1].

3.1.2 Grand Theft Auto V

The Playing for Benchmarks dataset, here denoted GTAV, consists of 1080p images sampled from video sequences from the video game *Grand Theft Auto V* (Richter et al., 2017). Each rendered image has information about the objects' labels and positions.

The training set consists of about 134,000 images which were collected on different time of day, in different weather conditions in the fictional city. Those images were here divided into 60/20/20% for training, validation, and testing, for the experiments[1].

The GTAV dataset consists of labels of objects that can be very far away or persons inside vehicles which makes them very hard or sometimes impossible to spot. Therefore, we filtered out small bounding boxes with an area smaller than 100 pixels. This area was chosen by empirical visual inspection of the ground-truth bounding boxes.

Furthermore, in the GTAV dataset, the hood of the driving car is labeled while it is not labeled in the BDD dataset. Therefore, we also removed the hood annotations from the dataset.

3.2 Intersection of Class Labels

The GTAV (Richter et al., 2017) and BDD (Yu et al., 2020) datasets use different class labels. GTAV has

[1] https://github.com/ljungqvistmartin/datasplits

32 classes while BDD has 10. Moreover, label names in the datasets also differ.

Therefore, a common subset of five classes was selected. This label space is called the *common* labels: car, person, cycle, truck, bus. The mapping from BDD and GTAV labels to the common labels are shown in Table 1.

3.3 YOLOv3

YOLOv3, *You Only Look Once version 3*, (Redmon and Farhadi, 2018) is a one-stage object detector. Compared to similar performing object detection methods, YOLOv3 claims to be faster at inference due to its one-stage detection process. The high inference speed is especially attractive in a real-time detection application.

The YOLOv3 architecture builds on extracting features from an image using Darknet-53, a backbone built of 23 residual blocks including 52 convolutional layers, which down-samples along the network depth using the stride length instead of max pooling.

The backbone is divided into residual blocks i.e. leveraging shortcut connections similarly to ResNet backbones (He et al., 2016). The benefit of such skip connections is that they deal with vanishing gradients and at the same time encourage feature reuse, which makes the model more parameter-efficient.

The YOLOv3 network contains 107 layers in total (numbered 0 to 106), of which 75 are convolutional layers, 23 residual (shortcut) layers (all in the backbone), 4 route layers where shortcuts end up (all in the head). Downscaling is done by a factor of two at layers 1, 5, 12, 37, and 62 in the backbone. Of the convolutional layers, 38 have a kernel of 3×3 and 37 have a kernel of 1×1.

The YOLOv3 network predicts bounding boxes at three resolution levels. These final prediction layers are referred to as detection layers; layers 82, 94, and 106. Each detection layer consists of a grid, where each cell contains the prediction of a bounding box, its objectness score, and a classification score for each class. All three detection layers are immediately preceded by seven convolutional layers.

After the low-resolution detection layer (layer 82), responsible for detecting high-level objects, the output is up-sampled (85) and concatenated (83 and 86) with intermediate output from Darknet-53 (61), which corresponds to the same up-sampled resolution. This concatenated tensor is passed through seven convolutional layers (87-93) and finally through the second detection layer (94). The same procedure is then repeated for the layers preceding the third and last detection layer (106).

The detection layers are grids, where the cells are responsible for predicting the bounding boxes as well as containing the predicted object and class probability. In inference, bounding boxes are non-maximum suppressed according to their objectiveness score, filtering out instances which the network believes have low probability of containing objects. The remaining bounding box predictions are then used in the actual prediction of the model.

3.4 The Models

CNNs are often divided into two parts: a backbone responsible for feature extraction and a detection head or classifier. Since training a backbone can be time consuming, training of CNNs often uses pre-trained backbone weights at initialization to reduce the computations needed. It is also advantageous for generalization.

The feature extraction layers could be considered general enough and that it is beneficial to freeze the layers as a kind of regularization (Hinterstoisser et al., 2018). On the other hand, the feature extraction layer weights may still have room for actual improvement and further training could increase the overall performance. Therefore, we analyze three differently trained models.

All trained models use the same hyperparameters: a learning rate of 10^{-4}, a batch size of 8, the Adam optimizer, 100 epochs with patience 10 (early stopping). Also, the random seed was set to the same value for all training sessions for them to have the same prerequisites, to be reproducible and reduce differences between models.

The images were scaled to 416×416 pixels for training, test, and analysis. However, the CKA comparison analysis was performed feeding images rescaled to 32×32 pixels to the networks to make the large matrices of concatenated activations fit in the working memory. Even though the models were not trained for this resolution, they have seen similar downscale resolution inside the network, but for a smaller input. The downscale inside the network will render correspondingly lower scale so each layer has not seen this particular scale at training. Touvron et al. (2019) have shown that for the convolutional part of a CNN the receptive field is unaffected by the input size. We focus on the similarity between the models and assume that the workings of the models are still viable.

There are multiple datasets with real and synthetic image data. For our experiments, we chose BDD to represent a dataset of real images, along with GTAV to represent a dataset of synthetic images.

Table 1: The label map between *BDD*, *GTAV* labels and the *common* labels.

Common	**BDD**	**GTAV**
person	person, rider	person
cycle	bike, motor	bicycle, motorcycle
car	car	car, van
bus	bus	bus
truck	truck	truck, trailer

All models were initialized with the ImageNet pre-trained Darknet-53 backbone which populates layers 0 to 74. Layers 75 up to layer 106 was populated randomly according to Kai-Ming uniform distribution.

U-Real – Further trained with all layers trainable (unfrozen) on our training set of BDD.

U-Synthetic – Further trained on the GTAV training set with all layers trainable (unfrozen).

F-Synthetic – Further trained on the GTAV training set with only detection head trainable i.e. layer 75-106 and thus leaving the backbone untrainable (frozen).

3.5 Similarity Metric

Comparing the similarity between two neural networks can be done in many ways. One approach is to look at the output for each individual layer and compare the outputs between networks. The problem can be described in the following way (Kornblith et al., 2019):

> Let $X_i \in \mathbb{R}^{p\times n}$ and $Y_i \in \mathbb{R}^{p\times n}$ be the output of layer i in form of matrices from two networks with p neurons each, fed with the same n inputs. We want to introduce a metric function $s(X_i, Y_i)$ that can be used to compare the similarity between two output matrices, to give insight of the behaviour and similarities between the hidden layers inside the models.

Several measures of similarity complying with this definition have been suggested. SVCCA (Raghu et al., 2017) and PWCCA (Morcos et al., 2018) are two examples of measuring representational similarity. Both metrics are invariant to invertible linear transforms i.e.

$$s(X,Y) = s(AX, BY) \tag{1}$$

for any invertible matrices A and B. This is argued to be an important property for comparing layer outputs. However, according to Kornblith et al. (2019), a metric with invariance to invertible linear transformations has the limitation of yielding the same similarity for all outputs with a greater width than the number of datapoints i.e. $p \geq n$.

The authors further argue that the scale of layer outputs also is important for representations. Therefore, similarity indices that preserve scale information, such as the Euclidean distance, can be helpful on giving insights of the activations. For a metric that is invariant to invertible transforms, the magnitude of the vectors in the activation space is irrelevant and therefore ignores this important information. Instead of requiring the similarity index to be invariant to invertible linear transform, a weaker invariance condition can be considered: invariance to orthogonal transformations. Invariance to orthogonal transformations means that $s(X,Y) = s(UX, VY)$ for any orthogonal matrices U and V. A property is that invariance to orthogonal transformations also means invariance to permutations which is important since the convolutional layer outputs should have the same representations independent of channel-wise permutations.

One such similarity index is linear CKA (Kornblith et al., 2019). CKA is not only invariant to orthogonal transforms but also invariant to isotropic scaling i.e. $s(X,Y) = s(\alpha X, \beta Y)$ for any $\alpha, \beta \in \mathbb{R}^+$. For the matrices X and Y, the CKA with a linear kernel is defined as:

$$CKA(X,Y) = \frac{||Y^T X||_F^2}{||X^T X||_F ||Y^T Y||_F}, \tag{2}$$

where $||\cdot||_F$ is the Frobenius norm and n is the number of data points i.e. columns in X and Y. With this index definition, Kornblith et al. (2019) have shown that the CKA captures intuitive similarity ideas such as models trained in the same way with different initialization should be similar.

In our experiments, we used linear CKA.

3.5.1 Convolutional Layers

While the CKA analysis requires matrices, the convolutional layers in the network output tensors. To solve this problem, we follow the line of Kornblith et al. (2019) and treat the output tensors of shape (n, h, w, c) as a collection of vectors of the shape $(n, h\cdot w\cdot c)$ where n is the number of images fed through the network, w and h are the width and height of the image, and c is the number of output channels (activations) i.e. the number of convolutional kernels for the specific layer.

3.6 Representational Similarity

Model U-Real gives us an indication of the performance we can obtain by only collecting a lot of real data.

Convolutional layers of the same layer index may have different roles in different networks trained on different data. Arguments can be made that the output of individual layers is not as important as the resulting output after a block of layers. However, here we focus on interpreting the single layer outputs.

The experiments used the CKA method described by Kornblith et al. (2019) to analyze the similarity between layers of several models.

The layer-wise similarity analysis was done by feeding 200 random images from our BDD test set through the trained networks and performing CKA on the layer outputs to find which layers are similar and which are not.

Residual layers i.e. shortcut layers essentially just sum outputs from two layers without any weights, they are though included in the CKA analysis for completeness of including all layers.

3.7 Implementation

The experiments were performed using the open-source implementation of YOLOv3 developed by Ultralytics (2019), using PyTorch 1.4 and the CKA implementation by Kornblith et al. (2019).

The performances presented as mean average precision (mAP) in the experiments are for all five common classes using mAP@0.5 i.e. mAP at 0.5 intersection over union (IoU).

4 RESULTS

In order to see that the trainings were successful, the resulting mAP of the trained models evaluated on our test set of the synthetic GTAV dataset and our test set of the real BDD dataset using image size 416 × 416 are presented in Table 2.

The models yielded best mAP on the type of data they were trained for, where U-Real got about 0.43 mAP on BDD while model U-Synthetic and F-Synthetic both only got about 0.12 mAP. Tested on GTAV model U-Synthetic and F-Synthetic both got about 0.89 mAP while U-Real got about 0.44 mAP.

It can be seen that model U-Synthetic and F-Synthetic had comparable mAP on both BDD and GTAV respectively, considering variations of trainings with different random seeds, see Table 2.

4.1 Layer-wise Analysis

The objective was to observe differences in models trained on real and synthetic data. Model U-Real was trained on real data only (ImageNet + BDD), while the head parts of U-Synthetic and F-Synthetic were trained on synthetic data only. Figures 1 and 2 show the results of the CKA similarity analysis using 200 images from our BDD test set that were fed through the models.

Summary statistics of all layer outputs (activations), averaged over all layers, are presented in Tables 3 and 4. A small difference in distribution can be observed between model U-Real trained on real data and the models trained on synthetic data: Models U-Synthetic and F-Synthetic. The difference was mostly in the mean and standard deviation. Comparably, models U-Synthetic and F-Synthetic have quite similar layer output value distribution. This can be observed both for image size 416 × 416 and 32 × 32, making it consistent between the mAP analysis and CKA analysis.

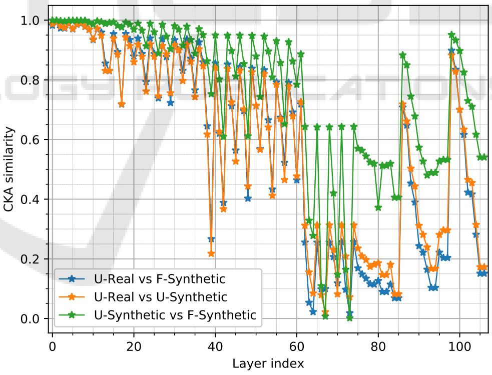

Figure 1: CKA similarity for all layers in entire YOLOv3 when images from our BDD test set were passed through the networks that were trained with seed 0.

The CKA similarity can be seen in Figures 1 and 2, where there was high similarity between real and synthetic models for both U-Synthetic and F-Synthetic in the first 13 layers of the network where all have similarity above 0.9 and most of them above 0.95. The similarity was above 0.7 for the first layers until layer 37. After layer 37 there was more variation in the similarity between the models. The similarity was quite high in most of the backbone until layer 61.

Comparing model U-Real with the synthetic models, the similarity from layer 62 to 85 was under 0.35.

Table 2: Performance of models trained on GTAV synthetic data: U-Synthetic and F-Synthetic as well as model U-Real trained on our BDD training set. All evaluated for mAP on our GTAV test set and our BDD test set.

Model	**mAP on BDD**		**mAP on GTAV**	
	seed 0	seed 1	seed 0	seed 1
U-Real	0.428	0.430	0.440	0.439
U-Synthetic	0.122	0.124	0.886	0.893
F-Synthetic	0.125	0.121	0.892	0.884

Table 3: Summary statistics of all layer outputs when feeding the network with 200 images of size 416 × 416 from our BDD test set. Values were averaged over all layers.

Model	seed	**mean**	**median**	**std**	**min**	**max**
U-Real	0	-0.0165	-0.116	0.961	-21.8	54.4
U-Real	1	-0.0145	-0.113	0.959	-21.0	52.1
U-Synthetic	0	-0.0272	-0.113	0.986	-22.0	52.4
U-Synthetic	1	-0.0269	-0.110	0.995	-22.3	50.3
F-Synthetic	0	-0.0263	-0.111	1.00	-22.2	51.0
F-Synthetic	1	-0.0235	-0.110	0.983	-23.2	48.0

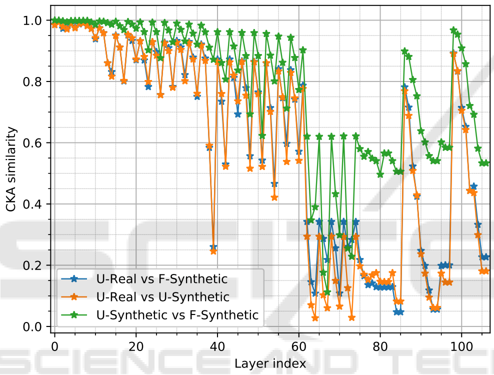

Figure 2: CKA similarity for all layers in entire YOLOv3 when images from our BDD test set were passed through the networks that were trained with seed 1.

The similarity was relatively low for the three detection layers in the head (layers 82, 94, 106), including their preceding layer, where all have similarity between 0.05 and 0.2.

In the head part, there were two peaks, at route layers 86 (concatenating the previous layer output with the output from layer 61) and 98 (concatenating the previous layer output with the output from layer 36), both routing from layers in the backbone. Since the similarity was high in the backbone overall, it is reasonable that there were similarity peaks where those two backbone layers are routed in the head part.

In the head part, each detection layer and the one immediately preceding convolutional layer had the same similarity values.

The average of CKA similarity was higher in the backbone than in the head part for both comparisons of U-Real vs the synthetic models, see Table 5. Likewise, the similarity between model U-Synthetic and model F-Synthetic was overall higher than when compared to model U-Real. The head part had lower similarity than the backbone and lower than the mean of all layers, for all comparisons.

Note the part between layer 62 and 85 in Figures 1 and 2 that all had lower values than the rest of the network in all comparisons.

The input images of size 32 × 32 have the size 1 × 1 in this region, which was lower than the convolutional kernel of 3 × 3 used in most layers in the entire network. However, for larger image sizes, it was not possible to perform these CKA calculations for the entire network since it would demand a vast amount of working memory. However, they could be performed for large parts of the network and larger image input sizes such as 128 × 128 showed similar patterns in that region, see Figure 7.

No difference was found for the U-Synthetic unfrozen model or the F-Synthetic frozen model in terms of overall average similarity with the unfrozen model U-Real, considering trainings with different random seeds, see Table 5. Thus, there was no overall impact of frozen or unfrozen in this regard.

4.2 Layer vs Layer Analysis

Looking at CKA similarity between layers shows how each layer compares to all other layers in the network, see Figures 3, 4, 5, and 6.

A row in these plots consists of the CKA similarity values between one layer in the model on the *y*-axis and all layers in the model on the *x*-axis.

A block-like structure was visible for different parts of the YOLOv3 architecture. Layers 0 to 12 were mostly similar to each other within the same model, as well as with other models, in all compar-

Table 4: Summary statistics of all layer outputs when feeding the network with 200 images of size 32 × 32 from our BDD test set. Values were averaged over all layers.

Model	seed	**mean**	**median**	**std**	**min**	**max**
U-Real	0	0.0613	-0.105	0.812	-16.8	36.8
U-Real	1	0.0634	-0.107	0.817	-17.0	33.8
U-Synthetic	0	0.0909	-0.103	0.743	-15.3	40.1
U-Synthetic	1	0.0989	-0.104	0.720	-15.8	37.3
F-Synthetic	0	0.0996	-0.105	0.726	-15.0	38.2
F-Synthetic	1	0.0925	-0.102	0.731	-15.2	39.5

Table 5: Mean CKA similarity for the models, for all layers, backbone and head.

Model	**all**		**backbone**		**head**	
	seed 0	seed 1	seed 0	seed 1	seed 0	seed 1
U-Real vs U-Synthetic	0.5865	0.5895	0.6925	0.7112	0.3379	0.3042
U-Real vs F-Synthetic	0.5734	0.6054	0.6931	0.7318	0.2930	0.3092
U-Synthetic vs F-Synthetic	0.7597	0.7845	0.8264	0.8461	0.6115	0.6508

isons. Blocks can be seen for layers 0 to 12, 14 to 37, 42 to 61, 62 to 74, 75 to 82, 83 to 94, and 95 to 106. This represents the architectural structure of YOLOv3.

As could be seen in the CKA plot in Figures 1 and 2, the impact of the routing layers (86 and 98) in the head part can be seen near the diagonal here. The part with the maximum downscale between layers 62 and 85 can be seen here as well, this part had lower similarity with most other layers in the network.

Figure 3 shows the similarity of all the layers against each other in model U-Real; self-similarity symmetric around the diagonal.

The diagonals of the plots of model U-Real vs U-Synthetic, U-Real vs F-Synthetic, and U-Synthetic vs F-Synthetic, seen in Figures 4, 5, and 6, are the same as the curves seen in Figure 1. The values off the diagonal thus show the similarity of layers with differing layer numbers.

The last layers in the backbone differs between comparisons of U-Real vs U-Synthetic, and U-Real vs F-Synthetic.

Most of the differences for real and synthetic were between layer 62 and 85, where the image was downscaled to the lowest scale. There U-Real seems more similar to layers in U-Synthetic and F-Synthetic, than U-Synthetic and F-Synthetic were similar to layers in U-Real.

Model U-Synthetic and F-Synthetic similarity to model U-Real for each layer can be seen in Figures 4 and 5. The similarity between all layers in model U-Synthetic (unfrozen) and F-Synthetic (frozen) can be seen in Figure 6. In comparison with the similarity plots of U-Real vs U-Synthetic, and U-Real vs F-Synthetic, the similarity between U-Synthetic and F-Synthetic was overall higher. There was high similarity in most of the backbone, specially the first part, even though model U-Synthetic had trainable backbone and model F-Synthetic had frozen backbone. However, a few layers in the backbone differs, for example the last layers in the backbone. The largest differences were thus in the head part, except for higher similarity around the route layers 86 and 98.

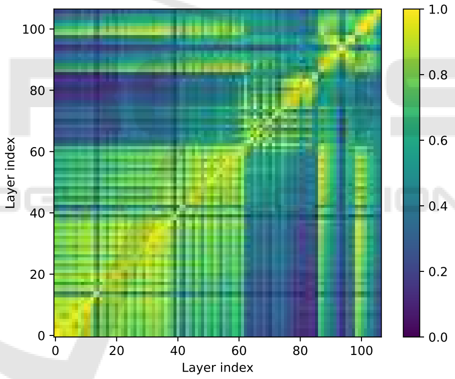

Figure 3: CKA similarity between layers of model U-Real that was trained with seed 0, for all layers in YOLOv3.

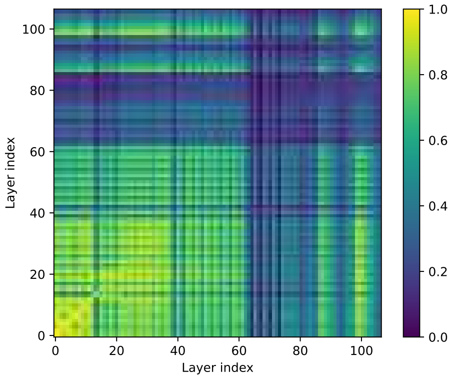

Figure 4: CKA similarity between layers of model U-Real (*y*-axis) vs model U-Synthetic (*x*-axis) that were trained with seed 0, for all layers in YOLOv3.

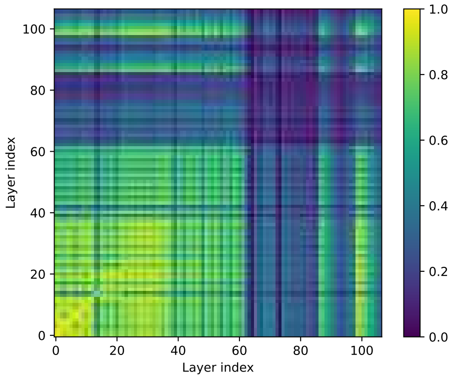

Figure 5: CKA similarity between layers of model U-Real (*y*-axis) vs model F-Synthetic (*x*-axis) that were trained with seed 0, for all layers in YOLOv3.

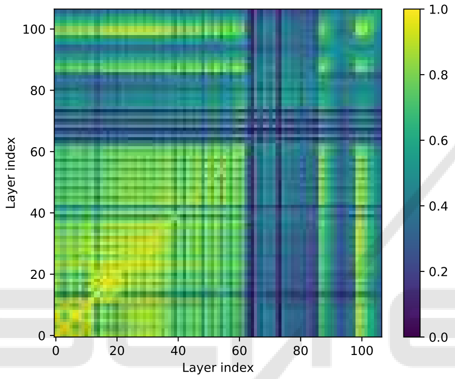

Figure 6: CKA similarity between layers of model U-Synthetic (*y*-axis) vs model F-Synthetic (*x*-axis) that were trained with seed 0, for all layers in YOLOv3.

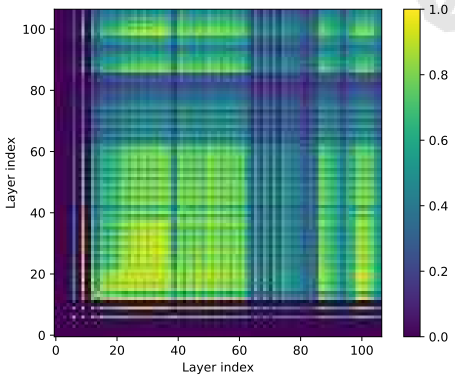

Figure 7: CKA similarity between layers of model U-Real (*y*-axis) vs model U-Synthetic (*x*-axis) that were trained with seed 0 and input size of 128 × 128. Results for layers 6, 9, 12 to 106 in YOLOv3.

5 DISCUSSION

The results overall showed small differences between model U-Real trained on real data and the models trained on synthetic data: Models U-Synthetic and F-Synthetic. The first part of the models showed high CKA similarity in all comparisons, while the head part showed more differences. All models had the same backbone pre-trained on real image data from ImageNet and model F-Synthetic did not have further training of the backbone. The high similarity between all models in the backbone means that the pre-trained backbone is rather dominant in all models, even after further training of the backbone in model U-Real and model U-Synthetic.

The first 13 layers in the backbone had very high similarity between all models, with similarity well above 0.9. Thus, the first layers in the network were not affected much by the dataset type and are likely mostly from the pre-trained backbone. These layers are likely targeting generic features. All models had the same backbone pre-trained on real image data from ImageNet, but that does not explain why the first 13 layers would be more similar than the rest of the backbone. The high similarity shows that the early layers of the different models develop similar representations, irrespective of if the dataset is real or synthetic.

The similarity between model U-Synthetic and model F-Synthetic was higher than when these models were compared to model U-Real. It seems like models trained on the same dataset develop similar representations. However, it could be further explored how much of this is due to real vs synthetic data and different datasets in general.

The F-Synthetic model with frozen backbone and the U-Synthetic model with unfrozen backbone, both further trained on the synthetic GTAV dataset, both had comparable mAP on BDD and GTAV respectively. No particular difference could be seen in the CKA analysis between frozen and unfrozen backbone. In Hinterstoisser et al. (2018), freezing the backbone during training on synthetic images yielded better performance on a real dataset compared to using an unfrozen backbone. However, Tremblay et al. (2018) showed promising results for unfrozen backbone. The diversity of the domain randomized dataset that they used could be the explanation to why they find differing results. To sum up, it seems that there is not a consensus whether freezing the backbone or not is preferred in all cases.

Comparing model U-Synthetic with unfrozen backbone and model F-Synthetic with frozen backbone, there were high similarity in most of the backbone between the two, specially the first part. However, a few layers in the backbone differ, for example the last layers in the backbone. Both models were derived from the same pre-trained backbone and perhaps the training of model U-Synthetic with trainable backbone did not result in large updates in the backbones.

In all comparisons, CKA similarity was lower than the rest of the network in the part between layer 62 and 85. The images used for CKA analysis were downscaled successively in the network and between layer 62 and 85 they have the smallest size. However, analysing larger image sizes show the same effect (see Figure 7) so the downscale cannot explain this solely.

The largest differences between model U-Synthetic and F-Synthetic were in the head part. Since model U-Synthetic had trainable backbone while model F-Synthetic had frozen backbone it would be expected that their backbones differ. Both networks were trained on the same detection task on the same dataset, so the head parts could likely become similar due to that. However, the head parts integrate information from multiple layers in the backbone that all have differences. Also, the receptive field increases with layer number and thus is quite large in the head part. These factors may explain why the largest differences were in the head part.

Kornblith et al. (2019) applied image classification on two different datasets with real images of resolution 32×32 using a 9 layer CNN network. The CKA similarity between the trained models was close to 1 for all comparisons for layers 1-4 irrespective of dataset, then dropped somewhat for later layers, especially after about layer 6. Similarity between the trained and untrained models was about 0.8 for the first layer and then dropped in a slope towards near zero for the last layer. This implies that a CKA similarity value of 0.8 could mean that the first layer of the trained model, which usually targets generic features, was somewhat similar to random noise. In another experiment with two untrained models with different initializations, the CKA similarity of the first layer was near 1 and for the first couple of layers were about 0.8 approximately. Our results are consistent with these results in that the early layers of the models showed high similarity, in our case above 0.9.

Higher CKA similarity values mean high similarity and vice versa, but in between high and low it is not entirely clear how different CKA similarity values should be interpreted.

Nguyen et al. (2021) showed further analyses of CKA on different ResNet architectures for image classification. They investigated the block structure of deep models, mainly ResNet. Since the backbone of YOLOv3 has similarities with ResNet, our analysis showed similar results on block structure.

Here we trained on image size 416×416 while analysing CKA on image size 32×32 which is a scale that the models were not trained for, which is a limitation, but we focus on the similarity between the models. Furthermore, in this work, one network architecture was analysed and trainings using one real image dataset with one synthetic image dataset were compared. In future work, the analysis would benefit of looking at multiple real and synthetic datasets and compare them as groups. Furthermore, different network architectures could also be analysed.

6 CONCLUSIONS

In our paper, we dissected models trained on real and synthetic images. We started from a backbone pre-trained on ImageNet real image data. Then:

- One model, U-Real, was further trained on real image data (BDD).
- Two other models were further trained on synthetic data (GTAV):
  - Model U-Synthetic with all layers trainable (unfrozen), and
  - Model F-Synthetic with a frozen backbone.

The trained models were evaluated on our test set of the synthetic GTAV dataset and our test set of the real BDD dataset. The trained models yielded best mAP on the type of data they were trained for.

Summary statistics of all layer outputs showed a small difference in distribution between model U-Real trained on real data and the models trained on synthetic data; models U-Synthetic and F-Synthetic. Comparably, models U-Synthetic and F-Synthetic have quite similar layer output value distribution.

The CKA similarity was calculated for comparing the model trained on real data, model U-Real, with models trained on synthetic data, models U-Synthetic and F-Synthetic. The average CKA similarity was higher in the backbone than in the head part when comparing the model trained on real data with the two models trained on synthetic data. Specially the first 13 layers in the backbone had very high similarity between all models, thus the first layers in the network were not affected much by the dataset type.

The similarity was quite high in most of the backbone until layer 61. From layer 62 to 85, the image size was the lowest and the similarity was relatively low.

The head part had lower similarity than the backbone, which was also lower than the mean of all layers. The similarity was relatively low for the three detection layers in the head.

Comparing CKA similarity values for layers vs layers showed a block-like structure resembling the different parts of the YOLOv3 architecture.

Model F-Synthetic with frozen backbone and model U-Synthetic with unfrozen backbone that were further trained on synthetic data had comparable mAP with each other, on both BDD and GTAV datasets. No particular difference could be seen in the CKA analysis between frozen and unfrozen backbone.

No difference was found for the U-Synthetic unfrozen model or the F-Synthetic frozen model in terms of average similarity with the unfrozen model U-Real. Thus, there was no overall impact of frozen or unfrozen according to CKA similarity.

The largest difference between model U-Synthetic and model F-Synthetic according to CKA was in the head part. Hence models U-Synthetic and F-Synthetic were more similar to each other in the backbone part than in the head part, even though their backbones had different training settings.

With this similarity analysis, we want to give insights on how training synthetic data affects each layer and to give a better understanding of the inner workings of complex neural networks. A better understanding is a step towards using synthetic data in an effective way and towards explainable and trustworthy models.

REFERENCES

Alain, G. and Bengio, Y. (2016). Understanding intermediate layers using linear classifier probes. In *ICLR 2017 workshop*.

Cabon, Y., Murray, N., and Humenberger, M. (2020). Virtual KITTI 2. *CoRR*, abs/2001.10773.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dosovitskiy, A., Ros, G., Codevilla, F., López, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16.

Fong, R. C. and Vedaldi, A. (2017). Interpretable explanations of black boxes by meaningful perturbation. In *The IEEE International Conference on Computer Vision (ICCV)*.

Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4340–4349.

Ge, Y., Xiao, Y., Xu, Z., Zheng, M., Karanam, S., Chen, T., Itti, L., and Wu, Z. (2021). A peek into the reasoning of neural networks: Interpreting with structural visual concepts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2195–2204.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*.

Golub, G. H. and Reinsch, C. (1971). *Singular Value Decomposition and Least Squares Solutions*, pages 134–151. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hardoon, D. R., Szedmak, S., and Shawe-Taylor, J. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664.

Hattori, H., Lee, N., Boddeti, V. N., Beainy, F., Kitani, K. M., and Kanade, T. (2018). Synthesizing a scene-specific pedestrian detector and pose estimator for static video surveillance. *International Journal of Computer Vision*, 126:1027–1044.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hermann, K. and Lampinen, A. (2020). What shapes feature representations? Exploring datasets, architectures, and training. In *Advances in Neural Information Processing Systems*, volume 33, pages 9995–10006. Curran Associates, Inc.

Hinterstoisser, S., Lepetit, V., Wohlhart, P., and Konolige, K. (2018). On pre-trained image features and synthetic images for deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*.

Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S. N., Rosaen, K., and Vasudevan, R. (2017). Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? In *IEEE International Conference on Robotics and Automation*, pages 1–8.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. E. (2019). Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Liu, T. and Mildner, A. (2020). Training deep neural networks on synthetic data. http://lup.lub.lu.se/student-papers/record/9030153. Master's Thesis.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing.

Morcos, A., Raghu, M., and Bengio, S. (2018). Insights on representational similarity in neural networks with canonical correlation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 5727–5736. Curran Associates, Inc.

Nguyen, T., Raghu, M., and Kornblith, S. (2021). Do wide and deep networks learn the same things? Uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations ICLR*.

Nowruzi, F. E., Kapoor, P., Kolhatkar, D., Hassanat, F. A., Laganière, R., and Rebut, J. (2019). How much real data do we actually need: Analyzing object detection performance using synthetic and real data. *ICML Workshop on AI for Autonomous Driving*.

Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. (2017). SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Redmon, J. and Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv*.

Richter, S. R., Hayder, Z., and Koltun, V. (2017). Playing for benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Ros, G., Sellart, L., Materzynska, J., Vázquez, D., and López, A. (2016). The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243.

Tan, M., Pang, R., and Le, Q. V. (2020). EfficientDet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Touvron, H., Vedaldi, A., Douze, M., and Jegou, H. (2019). Fixing the train-test resolution discrepancy. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Ultralytics (2019). Ultralytics implementation of YOLOv3. https://github.com/ultralytics/yolov3.

Wrenninge, M. and Unger, J. (2018). Synscapes: A photorealistic synthetic dataset for street scene parsing. *CoRR*, abs/1810.08705.

Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., and Darrell, T. (2020). BDD100K: A diverse driving video database with scalable annotation tooling. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham. Springer International Publishing.

Zhang, C., Bengio, S., and Singer, Y. (2019). Are all layers created equal? In *ICML 2019 Workshop Deep Phenomena*.