Escuela
Politécnica
Superior

Semantic Segmentation and Sim2Real

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Jordi Amorós Moreno
Tutor/es:
Nombre Apellido1 Apellido2 (tutor1)
Nombre Apellido1 Apellido2 (tutor2)

Mayo 2019

Universitat d'Alacant
Universidad de Alicante

Semantic Segmentation and Sim2Real

Subtítulo del proyecto

Autor
Jordi Amorós Moreno

Tutor/es
Nombre Apellido1 Apellido2 (tutor1)
Departamento del tutor
Nombre Apellido1 Apellido2 (tutor2)
Departamento del cotutor

Grado en Ingeniería Informática

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Mayo 2019

Preámbulo

Poner aquí un texto breve que debe incluir entre otras:

> "las razones que han llevado a la realización del estudio, el tema, la finalidad y el alcance y también los agradecimientos por las ayudas, por ejemplo apoyo económico (becas y subvenciones) y las consultas y discusiones con los tutores y colegas de trabajo."

Agradecimientos[1]

Este trabajo no habría sido posible sin el apoyo y el estímulo de mi colega y amigo, Doctor Rudolf Fliesning, bajo cuya supervisión escogí este tema y comencé la tesis. Sr. Quentin Travers, mi consejero en las etapas finales del trabajo, también ha sido generosamente servicial, y me ha ayudado de numerosos modos, incluyendo el resumen del contenido de los documentos que no estaban disponibles para mi examen, y en particular por permitirme leer, en cuanto estuvieron disponibles, las copias de los recientes extractos de los diarios de campaña del Vigilante Rupert Giles y la actual Cazadora la señorita Buffy Summers, que se encontraron con William the Bloody en 1998, y por facilitarme el pleno acceso a los diarios de anteriores Vigilantes relevantes a la carrera de William the Bloody.

También me gustaría agradecerle al Consejo la concesión de Wyndham-Pryce como Compañero, el cual me ha apoyado durante mis dos años de investigación, y la concesión de dos subvenciones de viajes, una para estudiar documentos en los Archivos de Vigilantes sellados en Munich, y otra para la investigación en campaña en Praga. Me gustaría agradecer a Sr. Travers, otra vez, por facilitarme la acreditación de seguridad para el trabajo en los Archivos de Munich, y al Doctor Fliesning por su apoyo colegial y ayuda en ambos viajes de investigación.

No puedo terminar sin agradecer a mi familia, en cuyo estímulo constante y amor he confiado a lo largo de mis años en la Academia. Estoy agradecida también a los ejemplos de mis difuntos hermano, Desmond Chalmers, Vigilante en Entrenamiento, y padre, Albert Chalmers, Vigilante. Su coraje resuelto y convicción siempre me inspirarán, y espero seguir, a mi propio y pequeño modo, la noble misión por la que dieron sus vidas.

Es a ellos a quien dedico este trabajo.

[1] Por si alguien tiene curiosidad, este "simpático" agradecimiento está tomado de la "Tesis de Lydia Chalmers" basada en el universo del programa de televisión Buffy, la Cazadora de Vampiros.http://www.buffy-cazavampiros.com/Spiketesis/tesis.inicio.htm

A mi esposa Marganit, y a mis hijos Ella Rose y Daniel Adams,
sin los cuales habría podido acabar este libro dos años antes [2]

[2]Dedicatoria de Joseph J. Roman en "An Introduction to Algebraic Topology"

Si consigo ver más lejos
es porque he conseguido auparme
a hombros de gigantes

Isaac Newton.

Contents

List of Figures

List of Tables

Listings

1 Introduction

1.1 Overview

In this bachelor's thesis we will be researching the sim2real field, specifically tackling the object detection problem from a semantic segmentation perspective. In the following chapters we will review and analyze the current state of art of some of the most important datasets, architectures and techniques used for semantic segmentation.

1.2 Motivation

The main motivation behind this project is to further investigate the sim2real and object detection field, specifically the UnrealROX project through semantic segmentation techniques, focusing in developing a user friendly framework for developers to easily generate synthetic data sequences in order to apply deep learning algorithms to real world environments.

1.3 Proposal and Goals

In this work we propose an extension to the UnrealROX project to automatize the synthetic data generation process as well as a study on how this data can be applied to semantic segmentation architectures to solve real world problems.

2 State of Art

As we previously stated, semantic segmentation is a extremely important task in the field of computer vision due to it's value towards complete scene understanding. In this section we will review some of the most common and recent deep network architectures, data-sets and frameworks that tackle the semantic segmentation problem.

2.1 Introduction

Before we dive into the next sections it is important to understand semantic segmentation and where it comes from. Semantic segmentation is a natural evolution of the object recognition problem, the goal is to infer the class for every pixel on the image, obtaining a pixel-by-pixel labeled image . Semantic segmentation is not so different from classic object recognition, in fact it's fundamentally the same, it just adds an extra layer of complexity towards the more fine-grained solutions. We could go further and try to differentiate instances of the same class, that would be instance segmentation.

image recognition grain figure

2.2 Sim 2 Real

In the last decade, data driven algorithms have vastly surpassed traditional techniques for computer vision problems, these algorithms, although can be tuned and improved in many different ways, still require vast amounts of quality, precisely annotated data in order to yield good results. In the real world environment, there are quite a few limitations to the quantity and quality of the data that can be produced. For instance, we could be limited to the number of cameras and annotators, moving physical objects to setup scenes could also be difficult and time consuming, and dangerous situations could be risky to set up properly i.e trying to get an autonomous car to learn to avoid pedestrians when there is no time to brake.

Sim2Real is a specific section of the data science field that mainly focuses on the automatic generation and ground-truth annotation of synthetic data by simulating the real world in a virtual environment. Although a virtual environment will allow us to workaround the previously mentioned restrictions, there's still a reality gap that must be covered in order for the synthetic data to be transferred to real life situations. Most synthetic data scenarios will present discrepancies between them and the real-world, to overcome this and properly transfer the knowledge to real problems there are two known approaches that have been proven to be effective. The first one and probably the most obvious is to generate extremely realistic environments, to achieve this multiple techniques can be applied, such as rendering very high and photo-realistic textures, models and lightning or simulating the noise of real cameras by applying filters and post-process effects. The other method, domain randomization Tobin et al. (2017) which is based on showing the model a huge amount of different synthetic data,

with this method even if the data is not extremely accurate to real data, the variability of the multiple range of data will make up for it and the model will learn.

In this section we will review some of the latest works on the field.

2.2.1 VirtualHome

VirtualHome is a three dimensional environment built in the Unity game engine. The main goal is to model complex tasks in a household environment as sequences of more atomic and simple instructions.

In order to perform this task a big database describing activities composed by multiple atomic steps was necessary, in the human natural language there is a lot of information that is common knowledge and is usually omitted, however, for a robot or agent this information has to be provided in order to fully understand. For this purpose an interface to formalize this tasks was built on top of the Scratch [1] MIT project. Then all of this atomic actions and interactions were implemented using the Unity3D game engine.

For the data collection, they had workers describe in natural language all of these tasks and then built them using the Scratch interface. Every task is composed by a sequence of steps where every step is a Scratch block, and every block defines a syntactic frame and a list of arguments for the different interactions that they may have.

Every step t in the program can be written as:

$$step_t = [action_t](object_{t,1})(id_{t,1})...(object_{t,n})(id_{t,n})$$

Where id is an identifier to differentiate instances of the same object. An example program to "watch tv" would look like this:

$step_1$ = [Walk] (TELEVISION)(1)
$step_2$ = [SwitchOn] (TELEVISION)(1)
$step_3$ = [Walk] (SOFA)(1)
$step_4$ = [Sit] (SOFA)(1)
$step_5$ = [Watch] (TELEVISION)(1)

2.2.2 UnrealROX

UnrealROX Martinez-Gonzalez et al. (2018) is a photo-realistic 3D virtual environment built in Unreal Engine 4 (UE4) capable of generating synthetic, ground-truth annotated data.

2.3 Common Architectures

As we previously stated, semantic segmentation is a natural step towards the more fine-grained image recognition problem, since the information that we are trying to infer is higher level, we will also require more complex architectures. Although the models we will be reviewing in this section work properly for image recognition and detection, some modifications will have to be made in order to adapt them for segmentation problems. However, they have

[1] `https://scratch.mit.edu/`

made such significant contributions to the field that they are still being used as the basic building blocks segmentation architectures.

2.3.1 AlexNet

AlexNet was the first deep network architecture that successfully surpassed traditional machine learning approaches, winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 with a 84.6% TOP-5 test accuracy, easily surpassing the its competitors by a huge margin. The architecture itself was pretty straight forward. It consisted of five convolution + pooling layers followed by three fully connected ones as seen in figure 2.1.

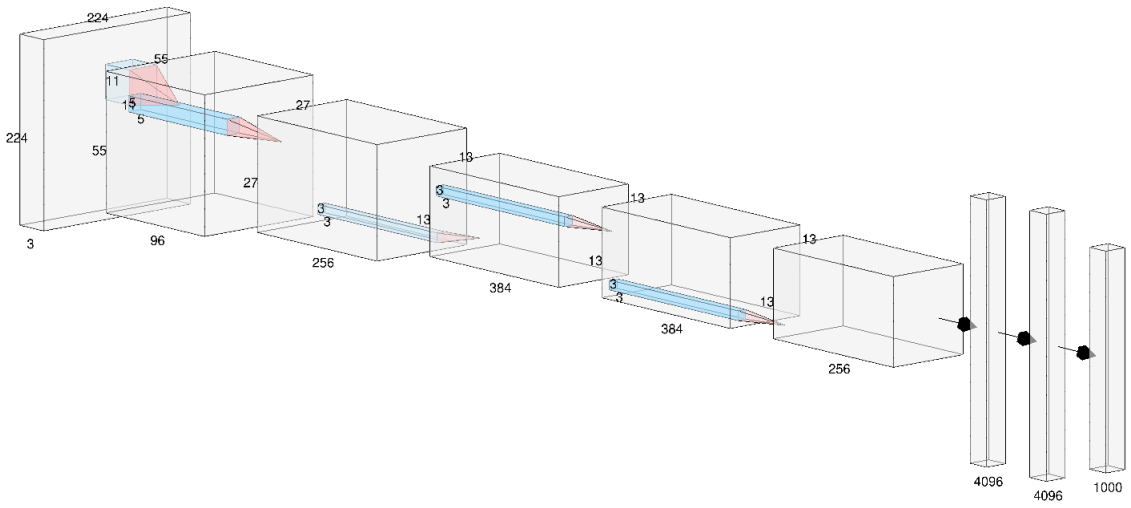

Figure 2.1: AlexNet architecture reproduced from Krizhevsky et al. (2012)

2.3.2 VGG

VGG is also a deep network model introduced by the Visual Geometry Group (VGG), one of the various model configurations proposed Simonyan & Zisserman (2015) was submitted to the ILSVRC 2013. The VGG-16 achieved 92.7% TOP-5 test accuracy.

2.3.3 GoogLeNet

GoogLeNet (also known as Inception) was introduced by (cite) and was submitted to ILSVRC 2014, winning with a TOP-5 test accuracy of 93.3%. GoogLeNet architecture is rather complex, it introduced the inception module (shown in figure 2.2) which consisted of a new approach where convolution layers were not stacked in just sequential order but instead had some of them compute in parallel, which substantially reduced computational cost.

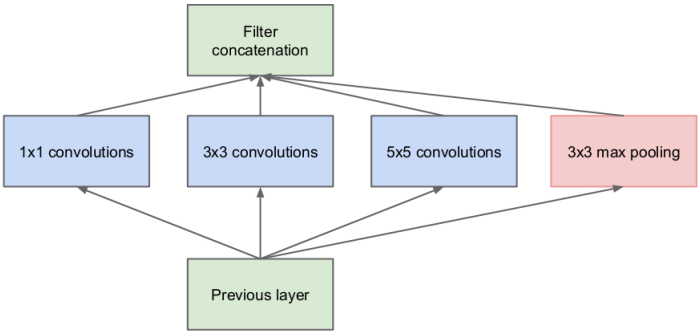

Figure 2.2: Inception module extracted from Szegedy et al. (2014)

2.3.4 ResNet

ResNet was first introduced by Microsoft research and it got very popular after achieving 96.4% in ILSVRC 2016. This Convolutional Neural Network (CNN) is extremely deep with 152 layers and it introduced a new concept called residual blocks (figure), this new module allowed the network inputs to skip layers and copy them to deeper layers, and so the compute output is a combination of X and F(X) as seen in (figure). This allows for deeper but also less computationally demanding networks.

2.3.5 ReNet

Multi Dimensional Recurrent Neural Network (MDRNN) are a variation of regular RNN that allow them to work with d spatio-temporal dimensions of the data. The architecture proposed by (cite) used regular RNN instead of MDRNN, where every convolution + pooling layer is replaced by four RNN's that sweep the image across in four directions.

2.3.6 Semantic Segmentation Methods

All the image recognition problem solutions based on convolutional architectures, whether it is recognition, detection, localization or segmentation, they all share a big common module, which is the convolution layers that will extract the features of any image, after that we can adapt our network for our specific problem.

Today, almost every semantic segmentation architecture uses the Fully Convolutional Network (FCN) by Long et al. (2014). The idea behind this is to replace the fully connected layers of previously well known architectures for image classification such as ResNet or GoogleNet with FCN in order to obtain a spatial map instead classification outputs, this way we can obtain pixel-wise classification while still using the inferred knowledge and power of the CNN's. Convolutional layers (convolution + pooling) downsample the input image in order to learn features, this downsampling does not matter when applied to classification problems, however, when doing pixel-wise classification, we require the output image to have the same size as the input. To overcome this problem, spatial maps are then upsampled by using deconvolution layers as shown in Zeiler et al. (2011).

2.3.6.1 Decoder Variant

The decoder variant is another method to adapt networks that were initially made for classification. In this variant, the network after removing the fully connected layers is normally called encoder and it outputs a low-resolution feature map. The second part of this variant is called decoder and the main idea behind it is to up-sample those feature maps to obtain a full resolution pixel-wise classification.

One of the most known examples of this encoder-decoder architecture is SegNet Badrinarayanan et al. (2015), the encoder part is fundamentally the same as a VGG-16 without the fully connected layers at the very end, while the decoder part consists of a combination of convolution and upsampling layers that correspond to the max-pooling ones in the encoder. SegNet is a very simple architecture wich yields very good results and is relatively fast, which makes it a good starting point of any semantic segmentation problem.

add segnet architecture fig

2.3.6.2 Dilated Convolutions

As we previously mentioned, CNN's generate significantly reduced spacial feature maps, to overcome this spatial reduction, dilated convolutions (also known as *atrous* convolutions) can be used in order to aggregate multi-scale contextual information without down-scaling.

The dilation rate can be modified by the upsampling factor. That way, a 1-dilated would be a regular convolution where every element has a receptive field of 1x1, in a 2-dilated every element has a 3x3 receptive field, in a 3-dilated every element has a 7x7 . This way the receptive field grows in a exponential way, while the parameters have a linear growth.

dilated conv fig

Some of the most important works that make use of this technique are the aforementioned multi-context aggregation by Yu & Koltun (2015) and DeepLab by Chen et al. (2016).

2.3.6.3 Conditional Random Fields

Deep CNNs applied to semantic segmentation excel at classification tasks, however they still lack precision when it comes to spacial information and struggle to properly delineate the boundaries of objects. To overcome this, a last post-processing step in order to refine the output can be applied, for instance, Conditional Random Fields (CRF). CRFs make use of both low level pixel interaction as well as the multi-class inference pixel prediction of high level models.

The DeepLab model by Chen et al. (2016) makes use of CRF to refine their output, figure shows the output of the CRF applied to the prediction output maps of their model.

deeplab crf fig

2.4 Datasets

In this section we will review some of the most important datasets that are commonly used to train semantic segmentation architectures.

2.4.1 PASCAL

PASCAL Visual Object Classes is one of the most popular 2D datasets for semantic segmentation. The challenge consists of 5 different competitions, 21 ground-truth annotated

classes and a private test set to verify the accuracy of submitted models. Also there are a few extensions of this dataset such as PASCAL Context or PASCAL Part.

2.4.2 Semantic Boundaries Dataset

This dataset is an extension of the PASCAL dataset that provides semantic segmentation ground-truth annotations for all the images that were not labeled in the original dataset. SBD greatly increases the amount of data from the original PASCAL and because of this is commonly used for deep learning architectures.

2.4.3 Cityscapes

Cityscapes is a urban dataset mainly used for instance and semantic segmentation. It contains over 25000 images and 30 different classes was recorded in 50 cities during different times of the day and year.

2.4.4 KITTI

The KITTI dataset Geiger et al. (2013) was recorded from a vehicle on a urban environment. It includes camera RGB images, laser scans, and precise GPS measurements. Despite being very popular for autonomous driving, it does not contain ground-truth annotations for semantic segmentation. To work around this, some researchers manually annotated parts of the dataset to fit their necessities.

3 Objectives

4 Materials and Methods

In this section we will go through some of the software and hardware specification related to this work, making special emphasis in those we ended up using as our main resources. We will divide this chapter in 4 different types of resources. First, we needed a game or 3D engine framework in order to generate our synthetic data. Second, we needed a high level deep learning framework in order to implement some of the current architectures, since most of them are quite complex and building them from the ground up is a extremely complex task and out of the scope of this work. Third, we need real world datasets in order to test our synthetic data trained algorithms. And finally, we need extremely powerful GPU computing in order to train and test these architectures.

4.1 UnrealEngine 4

UE4 is a very powerful, highly portable game engine, written in C++ and developed by Epic Games.

blueprints, class herarchy, editor

4.2 Frameworks

4.2.1 TensorFlow

TensorFlow is a open source library for numerical computation based on the idea of data flow graphs. In TensorFlow, the graph nodes represent the mathematical operations, while the edges represent the multidimensional data arrays (or tensors) flowing between them.

TensorFlow was created by the researches at Google Brain for the purpose of conducting machine learning and deep neural network research, its low level nature allows for a very fine-grained framework that can be use to build any architecture from the ground up and the tensor-graph structure also allows for very easy distribution on the CPU-GPU.

add flow graph figure

4.2.2 Keras

Keras is a high level framework written in Python that can use TensorFlow, CNTK or Theano as backend. It was developed with a focus on allowing for very fast experimentation and prototyping, abstracting the user of some of the more complex low level tasks with a very user friendly interface. This also makes Keras a very good entry framework for beginners that still don't have a solid foundation on deep learning.

Keras provides two different API's for different model building approaches. The Sequential API which allows to simply stack layer after layer, allowing for a very simple and easy to use interface for models with a input to output data flow. The Functional API however allows for more complex models by understanding each layer as a node graph, allowing for different, more complex and non sequential models.

4.2.3 PyTorch

PyTorch is an open source, Python-based computing package and machine learning framework

4.2.4 UnrealROX

5 Desarrollo

5.1 Expanding the UnrealROX Framework

As we previously mentioned on section 1 one of the main goals on this work is to expand the UnrealROX framework to easily generate synthetic data. In this section we will further detail UnrealROX framework and the data generation process.

UnrealROX can automatically generate and annotate data from a recorded sequence, but manually recording can be tedious and time consuming. In this work we have built the basic framework for the programmer to include its own actions and execute them in a sequential way, much like other frameworks like VirtualHome Puig et al. (2018).

5.1.1 The ROXBasePawn Class

This is the main class that contains all the logic for the character controller (movement, animations, grasping) of any robot pawn. It allows for the user introduce a robot to the scene and manually move and interact with the objects in a scene.

5.1.2 The ROXBotPawn Class

The ROXBotPawn class inherits from ROXBasePawn and will handle all the logic for the automation of tasks of any *Pawn* within a scene. In order to model all the different actions and interactions the Enum $EActionType$ was created, here the programmer can add any type of action to be built into the system.

Also in order to model the actions themselves the $FROXAction$ struct was built, containing a $AActor*$ target, as well as the type of action $EActionType$.

In order for the programmer to add actions and queue them from the UE4 editor we built the $doAction(AActor*, EActionType)$ and made it BlueprintCallable, this way, in a simple way actions can be queued from the editor and the *Pawn* will execute them in a sequential order as seen in figure 5.1.

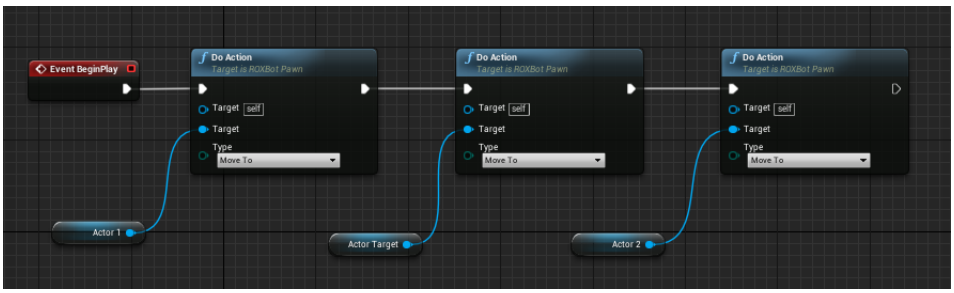

Figure 5.1: Example of queuing 3 "MoveTo" actions from the editor

The *doAction*() method creates a $FROXActions$ and pushes it to the queue. Whenever the queue contains an action and the *Pawn* is not executing one, it will run the $fetchNextAction()$ method. This will pop the action from the queue and execute the method corresponding to that action.

EQS and Naive MoveToActor methods of the UE4 Character class

Additional, pathfinding and movement logic was implemented in order to define a $MoveTo$ action. For this purpose the NavigationMesh component of UE4 was used along with the $FindPathToActorSynchronously$ method, which returns a $UNavigationPath$ containing all the path-points from one actor to another. Once we obtain the pathpoints the $VInterpConstantTo$ from the FMath library and the $RInterpTo_{C}onstant$ from the UKismeMathLibrary are used in order to obtain the next vector transformation for both position and rotation of the *Pawn*. These methods interpolate the current location with the next path-point location in order to achieve a smooth transition and make the movement more natural.

6 Results

7 Conclusions

Bibliography

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, *abs/1511.00561*. Retrieved from `http://arxiv.org/abs/1511.00561`

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2016). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, *abs/1606.00915*. Retrieved from `http://arxiv.org/abs/1606.00915`

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

Krizhevsky, A., Sutskever, I., & E. Hinton, G. (2012, 01). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, *25*. doi: 10.1145/3065386

Long, J., Shelhamer, E., & Darrell, T. (2014). Fully convolutional networks for semantic segmentation. *CoRR*, *abs/1411.4038*. Retrieved from `http://arxiv.org/abs/1411.4038`

Martinez-Gonzalez, P., Oprea, S., Garcia-Garcia, A., Jover-Alvarez, A., Orts-Escolano, S., & Rodríguez, J. G. (2018). Unrealrox: An extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation. *CoRR*, *abs/1810.06936*. Retrieved from `http://arxiv.org/abs/1810.06936`

Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., & Torralba, A. (2018). Virtualhome: Simulating household activities via programs. In *Computer vision and pattern recognition (cvpr)*.

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*. Retrieved from `http://arxiv.org/abs/1409.1556`

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., … Rabinovich, A. (2014). Going deeper with convolutions. *CoRR*, *abs/1409.4842*. Retrieved from `http://arxiv.org/abs/1409.4842`

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, *abs/1703.06907*. Retrieved from `http://arxiv.org/abs/1703.06907`

Yu, F., & Koltun, V. (2015). *Multi-scale context aggregation by dilated convolutions*.

Zeiler, M. D., Taylor, G. W., & Fergus, R. (2011, Nov). Adaptive deconvolutional networks for mid and high level feature learning. In *2011 international conference on computer vision* (p. 2018-2025). doi: 10.1109/ICCV.2011.6126474