# Context-Robust Object Recognition via Object Manipulations in a Synthetic 3D Environment


A THESIS PRESENTED
BY
DIMITAR KAREV
TO
THE DEPARTMENT OF COMPUTER SCIENCE
AND
THE DEPARTMENT OF NEUROSCIENCE


IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE JOINT DEGREE WITH HONORS
OF BACHELOR OF ARTS
IN THE SUBJECTS OF
COMPUTER SCIENCE AND NEUROSCIENCE


HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS

MARCH 2021

Neuroscience Concentration
Division of Life Sciences
Harvard University

## The Harvard College Honor Code

Members of the Harvard College community commit themselves to producing academic work of integrity – that is, work that adheres to the scholarly and intellectual standards of accurate attribution of sources, appropriate collection and use of data, and transparent acknowledgement of the contribution of others to their ideas, discoveries, interpretations, and conclusions. Cheating on exams or problem sets, plagiarizing or misrepresenting the ideas or language of someone else as one's own, falsifying data, or any other instance of academic dishonesty violates the standards of our community, as well as the standards of the wider world of learning and affairs.

*Signature: Dimitar Karev*

# Acknowledgments

The creation of this thesis would not have been possible without the generous help of many mentors and friends. First and foremost, I would like to thank Dr. Gabriel Kreiman for his guidance and for providing me with the opportunity to participate in research in his lab. Working in Kreiman Lab for the past year has taught me a lot about computer science, neuroscience, and what it means to be a researcher. Kreiman Lab has become an integral part of my college experience and I believe that it has provided me with skills that I will use for the rest of my life.

Next, I want to thank my daily lab mentor Dr. Mengmi Zhang for her constant help and support. Mengmi is not only an incredible researcher, but also a great mentor that has always been very patient and kind to me. I really appreciated all of our one-on-one meetings and I will be forever grateful for all of guidance and help that I have received.

I would also like to thank the other members on the team - Philipp Bomatter, Spandan Madan, and Claire Tseng. Each one of them has contributed a lot to this project and without their help, none of it would have been possible. They have always been very welcoming and friendly and I really enjoyed working with them.

Additionally, I would like to acknowledge the support of my academic advisor - Ryan Draft. Ryan has been an incredible mentor and a friend to me from the moment I stepped in the Biolabs as a confused sophomore. He was one of the first people to introduce me to the field of neuroscience and he was always there when I needed help with anything in my college career.

Finally, I would like to thank my friends and family who suffered through me constantly talking about my thesis. Thank you Galina, Kim, and Nish for helping me proofread and always encouraging me and helping me stay positive. Lastly, thank you, mom and dad, for supporting my academic endeavors and for providing me with unconditional love and support.

To everyone listed here, and to all of my professors, mentors, and friends at Harvard - thank you for making me who I am - this thesis would not have been possible without you!

# List of Contributions

Following the line of research connected to understanding various visual tasks in Kreiman Lab, the current research project was conceived by Dr. Gabriel Kreiman but designed and executed by Dimitar Karev, Dr. Gabriel Kreiman, Dr. Mengmi Zhang, Philipp Bomatter, Spandan Madan, and Claire Tseng.

The initial idea about using VirtualHome was given by Spandan Madan. He also contributed experimental design decisions and knowledge about Computer Vision and Machine Learning.

Generating the data with VirtualHome was done jointly by Dr. Mengmi Zhang, Dimitar Karev, and Claire Tseng. The generation of the "no context" images and the images violating gravity and/or the co-occurrence statistics was mainly done by Dr. Mengmi Zhang. The filtering of those images was jointly done by Dr. Mengmi Zhang, Dimitar Karev, and Claire Tseng. Dimitar Karev's other main contribution was to the generation and filtering of the images violating size regularities and the creation of a graphical user interface to assist with the image filtering processing.

The human psychophysics experiments were conducted by Dr. Mengmi Zhang. The analysis of the results presented in this thesis was jointly done by Dimitar Karev and Dr. Mengmi Zhang.

The novel computational model CRTNet was conceived by Dr. Mengmi Zhang and Philipp Bomatter and it was implement by Philipp Bomatter. Dimitar Karev adapted the data from Zhang et al., 2020 discussed in Section 4.2, and trained and tested all of the computational models (Faster R-CNN, DenseNet, CATNet, and CRTNet) on it. Philipp Bomatter trained and tested the computational models on the new data from VirtualHome.

The analysis of the results presented in the thesis was mainly done by Dimitar Karev with the help of Dr. Gabriel Kreiman, Dr. Mengmi Zhang, and Philipp Bomatter.

Thesis advisor: Professor Gabriel Kreiman Dimitar Karev

# Context-Robust Object Recognition via Object Manipulations in a Synthetic 3D Environment

## Abstract

The remote control is a small object that does not fly in the air and is generally found on a table, not in the sink. Such contextual regularities are ingrained in our perception of the world and previous research suggests that they can even influence human and computational models object recognition ability. However, the exact effects of contextual information on object recognition are still unknown for both humans and machine learning models. Here, we introduce a novel way of studying the effects of different contextual cues in a qualitative and systematic way. We present a diverse synthetic dataset created via a 3D simulation engine that allows for complex object modifications. Our dataset consists of more than 15000 images across 36 object categories and it is designed specifically for studying the effects of gravity, object co-occurrence statistics, and relative size regularities. We conduct a series of psychophysics experiments to assess human performance and establish a benchmark for computational models on the dataset. Additionally, we test state-of-the-art deep learning models on the same dataset and study how contextual information influences their object recognition accuracy. Finally, we propose a context-aware recognition transformer network that integrates contextual and object information via multi-head attention mechanism. Our model captures useful contextual information that allows it to achieve human-level performance and significantly better robustness in out-of-context conditions compared to baseline models across our dataset and another existing out-of-context natural image dataset. Moreover, our model performs in a way that is consistent with human object recognition and shows similar recognition artefacts.

# Contents

# 1

# Introduction

When you see a remote control, how do you know that you are looking at a remote control? Visual object recognition in its general form is computationally an incredibly hard task. Still, humans are extremely good at telling objects apart in all sorts of lighting conditions, on different backgrounds, and from various distances and angles. Object recognition feels instantaneous and effortless: it takes less than 300 ms to assimilate an image and happens involuntarily (Potter, 1976) (Thorpe et al., 1996). It is no surprise that such complicated and well optimized behavior is still an open problem

for both computer science and neuroscience.

In the rest of this section, we hope to (1) give a brief summary about what we know about object recognition in the brain, (2) offer a historical overview of the problem from a computer vision perspective, and (3) discuss some of the limitations of modern object recognition architectures and give a short summary of our approach to making a context-robust model.

## 1.1 Object Recognition in the Brain: The Visual System

Before diving into the details of how the brain recognizes objects, it is worth noting that the visual system performs many different tasks beyond object recognition. These tasks do not happen in isolation of each other and they very often are needed as prerequisites for one another – e.g. light photons need to be registered by the brain, the brain then identifies all of the objects in an image, it determines their sizes, relative positioning, movement etc., before recognizing the objects themselves (Sáry et al., 1993) (Adelson and Bergen, 1985) (Logothetis and Sheinberg, 1996). Therefore, here we will define object recognition specifically as the task of naming a single given object in an image. Note that we will assume that the boundaries of the object are already known, forming a distinction from tasks such as image detection (detect and identify any object in an image) and image captioning (describe what is happening in an image).

Some of the first more complete descriptions of the visual system come from Felleman and Van Essen and their famous experiments on macaque monkeys (1991). Felleman and Van Essen describe several different areas in the visual system organized in a hierarchical manner. All of these areas can be roughly split into two parts: the ventral stream, known as the "what" pathway, and the dorsal stream, known as the "where" or "how" pathway (Fig. 1.1). The difference between these streams can be explained by the "perception/action" dichotomy (Goodale and Milner, 1992). The ventral stream is often associated with perception of the stimulus, and thus it is involved in repre-

senting and recognizing objects. The dorsal stream is responsible for more "action-based" tasks such as avoiding an obstacle, eye movements, and grasping the object (Jeannerod et al., 1995). Note, that while we often talk about the ventral and dorsal streams as two distinct and independent areas, they still communicate with each other and recent evidence suggests that the connection between the streams is involved in complex object-oriented hand movements (van Polanen and Davare, 2015).

While we still do not know everything about the visual system, we have good hypotheses about the specific functions of the small areas within it. Kruger et al. (2012) summarize the role of each area in the visual system. The photoreceptors in the retina pass information about the image to the retinal ganglion cells and the lateral geniculate nucleus. The latter two areas play multiple roles in processing information – cells there detect both spatial and temporal changes and therefore they are often described as edge detectors. Other cells play the role of low-pass filters which helps the robustness of the system by reducing noise (Merigan and Eskin, 1986). After the retinal ganglion cells and the lateral geniculate nucleus, the information is forwarded to the primary visual cortex (V1) where neurons respond to a wider variety of stimuli such as more edges, motion, color, depth, etc. The next area is V2 which has all of the features seen in V1 but also adds contour representation which helps with image segmentation. After V2, the visual system splits into the ventral and dorsal pathways which eventually end in the inferior temporal areas and in the intraparietal areas respectively (Kruger et al., 2012).

In order get a better insight about how the brain resolves object recognition, we need to concentrate on the ventral pathway. Even though we have a relatively good understanding of lower visual processing areas (retinal ganglion cells, lateral geniculate nucleus, V1, and V2), our understanding of higher stages still remains limited. We know that V4 combines inputs mostly from V2 to create more complex stimuli which are harder to interpret. It has been hypothesized that V4 and higher areas encode invariant representations which helps us recognize objects in various conditions (Orban, 2008). One example of such invariance is luminance invariance in V4 – color coding cells in V4

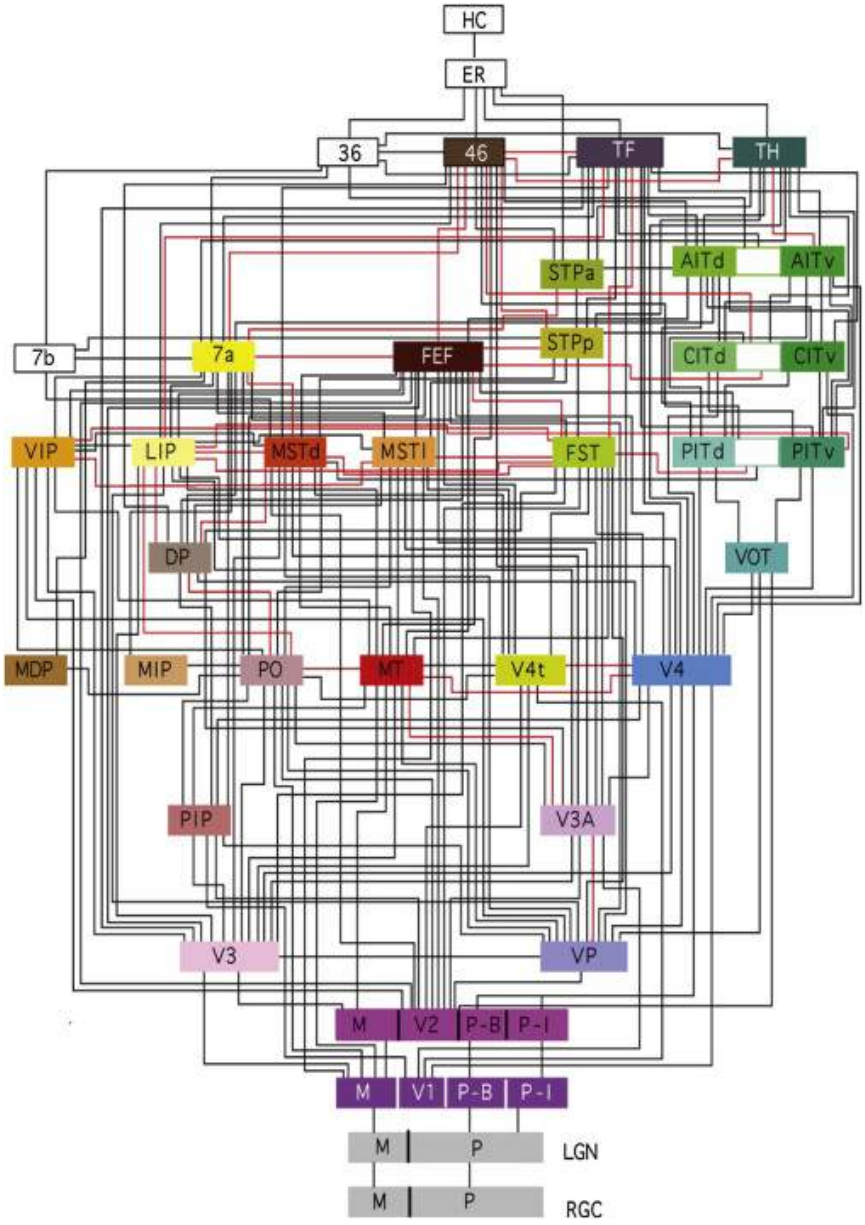



**Figure 1.1: Hierarchical map of the visual system in macaque monkeys.** Taken from Felleman and Van Essen, 1991. Visual input is at the bottom, starting with retinal ganglion cells (RTC) and lateral geniculate nucleus (LTN); the visual system culminates at the top where it makes connections with the entorhinal cortex (EC) and the hippocampus (HC). The ventral stream is depicted on the right, ending in the inferior temporal (IT) areas (PIT, CIT, AIT). The dorsal stream is depicted on the left, ending in the intraparietal (IP) areas (LIP, VIP).

are different than color coding cells in V2 since they respond to hue rather than color opponency (Rentzeperis and Kiper, 2010). This invariance representation allows the brain to concentrate on important features of the image and generalize better when making object recognition decisions in the later stages of visual processing – a dark green lime is still a lime but a yellow lime might look more similar to a lemon.

Invariant coding is even more prominent at the final stages of visual processing – the inferior temporal (IT) areas. Historically, the IT has been seen as being composed of two parts: the posterior IT (also known as TEO) and the anterior IT (also known as TE)(von Bonin, 1947) (Boussaoud et al., 1991) (Saleem et al., 2000). Some researchers have also considered parsing the IT into three parts and adding a central IT area (Felleman and Van Essen, 1991) (Fig. 1.1). However, new evidence shows that most likely the IT is comprised of four distinct functionally biased regions (Kravitz et al., 2013) (Conway, 2018). Determining the exact structure of the IT has proven to be a very hard task. The neurons in this region have much more complex activation stimuli than V4 and the overall processing in the area is still not well understood (DiCarlo et al., 2012). Nevertheless, there is some evidence showing that cells in IT are activated by a wide range of stimuli from simple oriented 2D shapes (Brincat and Connor, 2004) to more complicated 3D objects (Yamane et al., 2008) and even forms similar to hands and faces (Tanaka, 1996). More strikingly, it has been shown that neurons preserve their activation preference over changes in luminance, clutter, position, size, etc. (Kreiman et al., 2006) (Brincat and Connor, 2004), (Rust and DiCarlo, 2010), (Zoccolan et al., 2005), (Vogels and Biederman, 2002).

In order to perform object recognition, the visual system has to complete two seemingly contradictory requirements: selectivity and invariance (Kruger et al., 2012). The visual system needs to respond to small differences in stimuli, so that it can differentiate between objects and detect small changes in object appearance. At the same time, the visual system needs to represent many invariant features, so that it can classify seemingly different visual stimuli as the same object. What makes the

visual system so complex and interesting is not necessarily its ability to perceive different images but the added ability to treat completely different visual stimuli like changes in object's size, color, position, illumination, etc. as the same object. All of this is achieved by neurons in V4 and in the IT that respond to invariant stimuli.

All of these findings support the hypothesis that objects are represented in a distributed manner in the brain, meaning that no single neuron is responsible for recognizing an object but it is the combined activation of a group of neurons. Additionally, previous research has found that the IT is sufficient to support core object recognition (Hung et al., 2005). Some results indicate that very simple statistical classifiers such as support vector machines can be trained to recognize objects only based on responses from IT neurons (Kiani et al., 2007) (Tompa and Sáry, 2010) (Rust and DiCarlo, 2010). All of these neuronal responses have been shown to be present in passively looking subjects that received no explicit training and therefore are likely the neural correlates for object recognition (Hung et al., 2005).

Note that in our discussion so far, we have simplified many of the complications of the visual system. We talked about the ventral stream as a stream of feed-forward connections, starting at V1 and ending at IT, but that is not necessarily the case. There is much evidence suggesting that there are feedback connections between all different visual areas (feedback connections are not made only on the retina) (Bullier, 2001) (Roelfsema et al., 2000). New research shows that feedback computations might be highly advantageous in areas like perceptual groupings based on long-range spatial dependencies, visual reasoning, generalization, plasticity, and others (Kreiman and Serre, 2020). Furthermore, not every connection is made between two "neighbouring" areas. There is evidence suggesting that there are also some number of "jumping" connections between V2 and IT (Nakamura et al., 1993) and between V4 and IT (Saleem, 1992). These features of the visual system are still poorly understood and more research is needed to investigate their role in object recognition.

To summarize, the visual system consists of ventral and dorsal streams, but it has been hypoth-

esized that the ventral stream plays a more crucial role in object recognition. The ventral stream consist of V1, V2, V4, and IT, and each area processes the visual information in a unique way before passing it to the next. Object recognition requires neurons to compute invariant features, which is a non-trivial task still not well understood by scientists. Since early visual neurons do not exhibit invariance to the same extent as neurons in the IT, it is postulated that IT cells solve the problem of object recognition in a distributed manner.

## 1.2 Object Recognition in the Computer: Machine Learning

The field of computer science has always been fascinated with the task of object recognition. One of the first efforts in designing a computational model for the purposes of image recognition was done as early as 1958 with the invention of the perceptron. The perceptron was first designed as a mathematical model of the biological neuron (McCulloch and Pitts, 1943). The pioneering neural architecture consisted of a single layer of output nodes connected with weighted edges another layer of input nodes. Constructed for image recognition, it acted as a binary classifier based on a linear activation function. It was the first *supervised learning model* – that is, an architecture that improves its performance based on labeled input data (Rosenblatt, 1958).

Even though the perceptron on its own had very limited learning capabilities, it gave the foundations on which the field of machine learning is standing. The invention of backpropagation (Rumelhart et al., 1986) allowed scientists to combine multiple layers of neurons together and train them on all sorts of tasks including object recognition.

In 1989, Yann LeCun demonstrated the power of backpropagation by applying it to the problem of recognizing handwritten zip codes (LeCun et al., 1989). In his paper, LeCun used covolutional networks which played a key role in recognizing the handwritten digits. Inspired by receptive fields in the primate visual system (Fig. 1.1), convolutional neural networks (CNNs) have been used to

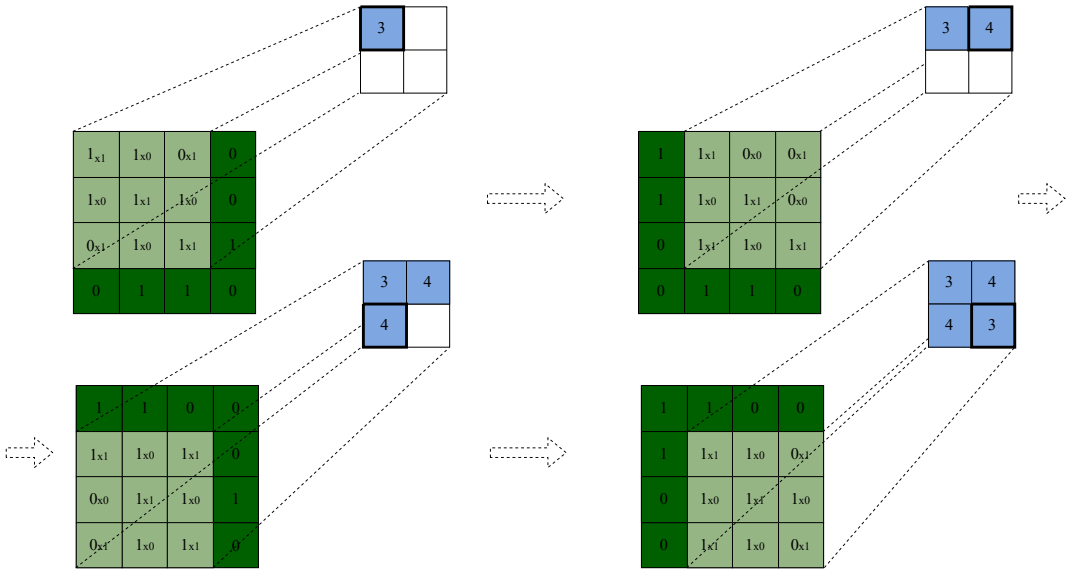


**Figure 1.2: An example of a convolutional layer.** A convolution is performed with a kernel of size $3 \times 3$ on an input of size $4 \times 4$.

capture spatial and temporal dependencies in an image. They work by applying a number of filters on the target image that extract high-level information such as the presence or absence of edges, shapes, and patterns.

Convolutional networks typically have pooling layer and convolutional layers. Pooling layers can be max pooling, average pooling, ROI pooling (e.g. see Fig. 2.1), etc., and they are usually used for decreasing computational power and/or extracting dominant features. Pooling layers do not have weights and they are not trainable. Convolutional layers (Fig. 1.2) on the other hand can be trained. They are mainly used for feature extraction and finding patterns (Albawi et al., 2017). Nowadays, both pooling layers and convolutional layers are used in every object recognition model due to their powerful feature extraction capabilities and fast computations. However, their full capabilities were not fully utilized by machine learning models until 2012, when AlexNet (Krizhevsky et al., 2012) won one of the biggest object recognition annual competitions – ImageNet Large Scale Visual Recognition Challenge (Russakovsky et al., 2015) by a large margin.

AlexNet is sometimes considered to be the founding network of modern object recognition (Alom et al., 2018) due to its revolutionary for its time approach to the problem. The network was inspired by more than a decade-old LeNet (LeCun et al., 1998) that was one of the first networks to use layers of CNNs to extract a feature map from an image. AlexNet build off LeNet by introducing more CNNs and thus making the network even deeper and wider. To be able to train such big network efficiently, the authors used a Graphic Processing Units (GPU) implementation which started the everlasting practice of using GPUs for training machine learning models. This combination of using a big number of convolutional layers in a network trained on a GPU proved to be very successful as AlexNet achieved lower error rate than any state-of-the-art model at the time (15.3% versus 26.2% (second place) error rates).

More specifically, the architecture of AlexNet consists of 5 convolutional layers followed by 3 fully-connected layers (see Fig. 1.3). The first 2 convolutional layers are also followed by local normalization and max pooling layers, whereas the first 2 fully-connected layers are used with dropout. The last layer is a softmax layer. All layers are followed by ReLU non-linearity (Krizhevsky et al., 2012). Due to using GPUs for training, AlexNet was also able to be much larger than other networks at the time. The total number of parameters in AlexNet is on the order of 61M and the total number of connections is over 600M (Alom et al., 2018).

It is safe to say that AlexNet had a big impact on the field of object recognition and machine learning as a whole. Currently, most machine learning models designed for any task use at least some number of convolutional layers and are typically trained on a GPU. Most state-of-the-art architectures also have much more than 61M parameters and 600M connections (Alom et al., 2018).

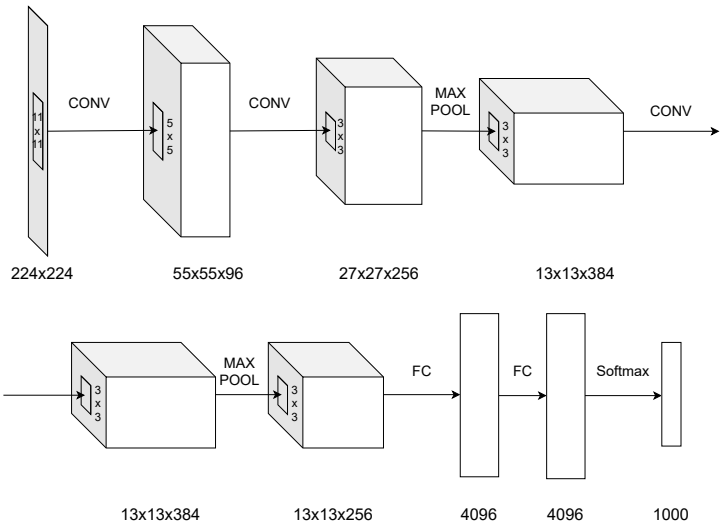



**Figure 1.3: A diagram of the architecture of AlexNet.** AlexNet consists of 5 convolutional layers, followed by 3 fully-connected layers. The diagram shows all of the layers and the size of the feature map after every layer.

## 1.3 Out-of-context Object Recognition

Some of the more popular architectures that were designed after AlexNet include the R-CNN family (Girshick et al., 2014) (Girshick, 2015) (Ren et al., 2015), VGGNet (Simonyan and Zisserman, 2014), the YOLO network (Redmon et al., 2016), ResNet (He et al., 2016), and DenseNet (Huang et al., 2017). While all of these networks have proven themselves to be incredibly powerful tools for recognizing objects, they still have not matched human recognition performance on a variety of tasks. Additionally, all of these networks have been shown to produce many non-human like errors – like being fooled by changes of a few pixels in an image (Serban et al., 2020) or by changes in the contextual information of an image (Rosenfeld et al., 2018) (Zhang et al., 2020).

Such mistakes happen due to the fact that neural networks are incredibly powerful association machines. They learn co-occurrence statistics not only between the object's appearance and its label but also between the object's background and its label (Divvala et al., 2009) (Sun and Jacobs, 2017)

(Beery et al., 2018). Previous research has shown that context plays a major role in object recognition in both humans and machines (Torralba et al., 2003) (Choi et al., 2012) (Zhang et al., 2020). Deep networks trained on natural image datasets like ImageNet (Krizhevsky et al., 2012) have been demonstrated to rely strongly on context (Geirhos et al., 2018), (Brendel and Bethge, 2019), (Singh et al., 2020). While, intuitively relying on context might seem beneficial due to having more visual information to work with, when a model relies too much on context it can be detrimental to its performance. Indeed, research shows that such models often fail when objects are placed in an incongruent context (Rosenfeld et al., 2018) (Zhang et al., 2020).

Most work in the literature has represented context in an oversimplified way - just as the background of an image. This includes testing the generalization to new backgrounds (Beery et al., 2018), incongruent backgrounds (Zhang et al., 2020), exploring impact of foreground-background relationships on data augmentation (Dvornik et al., 2018), and replacing image sub-regions by another sub-image i.e. object transplanting (Rosenfeld et al., 2018). However, we still do not understand how other more complex contextual irregularities affect object recognition. To the best of our knowledge, there is no existing work exploring aspects of object context such as gravity and object size regularities.

Here we develop a novel way to systematically and quantitatively study the effects of context on object recognition. We leverage the power of a 3D simulation engine that allows for complex object manipulations to create a new out-of-context image dataset. The dataset is designed to help us rigorously study the effects of 6 contextual conditions - normal context, gravity violation, object co-occurrences violation, size regularities violation, combination of gravity and object co-occurrences, and no context.

We used the generated images to gain insight into how these particular contextual irregularities affect object recognition performance in both humans and state-of-the-art computer vision models. To do that, we conducted a series of human psychophysics experiments which allowed us to

establish a benchmark for the task. We also tested three state-of-the-art machine learning models and assessed the effects of context violations on these models.

Finally, in an effort to create a more biologically plausible neural architecture that produces more robust object recognition predictions, we propose the Context-aware Recognition Transformer Network (CRTNet). CRTNet consists of two streams - a context stream and a target stream. The context stream processes and integrates contextual and target cues via multi-head transformer decoding layers. The target stream processes only the target object and uses a confidence estimator to combine the information from both streams.

# 2

# Related Work

In this section, we will review some of the state-of-the-art object recognition models that will also appear later in our analysis in constructing baseline performance. Namely, we will review Faster R-CNN (Ren et al., 2015), DenseNet (Huang et al., 2017), and CATNet (Zhang et al., 2020). We chose these models because of their high levels of acceptance and impact on the field (Faster R-CNN and DenseNet have been cited more than 14000 times each) or their direct applicability to the task of out-of-context object recognition (CATNet has shown promising human-like object

recognition features).

The goal of this section is not teach the reader everything there is to know about Machine Learning nor to present all of the details connected to these state-of-the-art models. This would have taken many pages and it is not in the scope of this senior thesis. Instead, we hope to provide the reader with some context about what state-of-the-art object recognition architectures look like and how their shortcomings have informed the design of our novel model introduced in Section 3.

## 2.1 Faster R-CNN

One of the most popular and best performing deep learning architectures is Faster R-CNN. It was designed in 2015 inspired by a series of "Region-Based Convolutional Neural Networks" (R-CNNs) (Ren et al., 2015). The most popular members of the R-CNN family include R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), and Faster R-CNN (Ren et al., 2015), and as the names suggest the evolution between versions has been mainly concentrated on improving computational efficiency.

Here we will quickly describe the structure of Faster R-CNN and explain how we modified it for the purposes of the project. Faster R-CNN consists of the following regions:

1. Region Proposal Network: A region that generates plausible segmentations of the image identifying the location of different objects within it. This stage is used for object detection (i.e. when the location of the object of interest in the image is unknown).

2. CNN for Feature Extraction: A region that processes the different parts of the image and generates features used for recognizing the object. This is the main part of the network responsible for object recognition.

3. Classification Stage: A final stage used to predict the label for the target object and/or im-

prove bounding box prediction.

Note that Faster R-CNN was first designed for the task of object detection, so in its original version it contains a network that allows for generating guesses about the location of the objects in the image (i.e. generating bounding boxes). However, for the purposes of this project, we will ignore that stage and will assume that the location of the object of interest is known to us. Thus, our implementation of the model replaces the region proposal network with the ground truth bounding box that comes with our data (Fig. 2.2).

A crucial part of object recognition is the process of feature extraction. Just like all other state-of-the-art models, Faster R-CNN achieves this feature extraction through the use of a convolutional neural network. The architecture first resizes the input image to $1000 \times 600$ and then it feeds the resized image into a pretrained deep CNN such as VGG-16 (Simonyan and Zisserman, 2014) or ZF-Net (Zeiler and Fergus, 2014). The CNN, in the case of VGG-16, extracts features of the image through a series of 13 convolutional layers with stride 1 and 4 max pooling layers with stride 2. The end result of the feature extraction is a feature set of size $60 \times 40 \times 512$ (since the maxpooling layers half the size of the image every time and there are 512 channels in the last convolutional layer (Simonyan and Zisserman, 2014)). This step is crucial since it provides the network with plenty of useful features that can be used for object recognition.

Next, the feature set is forwarded to *Region of Interest* (ROI) pooling layer. The ROI pooling layer works by taking the bounding box corresponding to the target object and dividing it into a fixed number of roughly equal sized regions (in the case of Faster R-CNN that is 49 regions organized in a square $7 \times 7$). Max pooling is performed to obtain the largest number from each region. Thus, we obtain an output of $7 \times 7 \times 512$ from the ROI pooling layer. This stage has proven to be incredibly useful for the network since it reduces the computational complexity of the model and allows it to concentrate on the target object. For an example of ROI pooling, see Fig. 2.1.

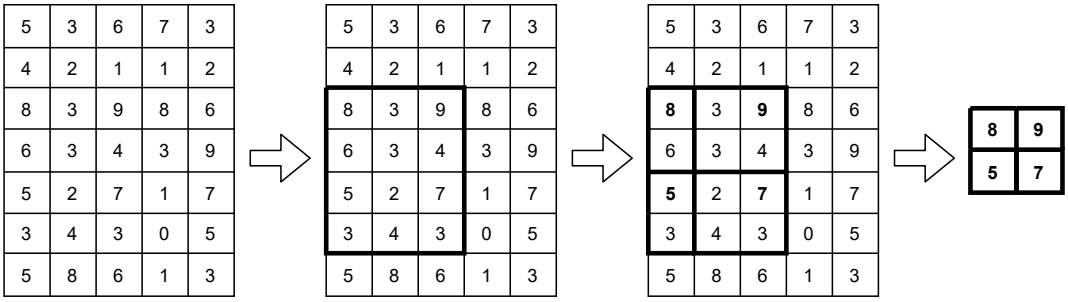



**Figure 2.1: An example of ROI pooling.** An example of ROI pooling that takes $7 \times 5$ table and outputs $2 \times 2$ table. The bolded cells represent the region of interest (i.e. the location of the target object in Faster R-CNN) over which we are performing max pooling.

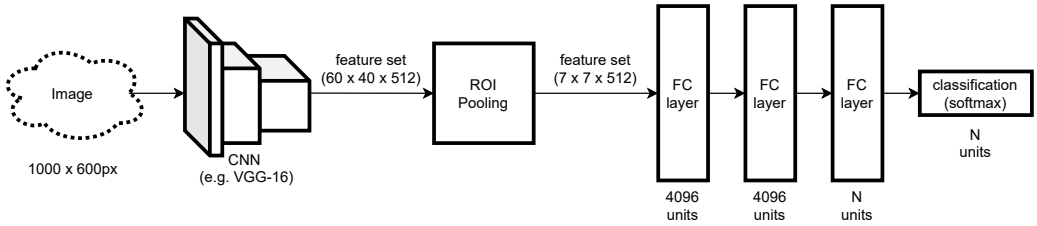



**Figure 2.2: A diagram of our modified Faster R-CNN.** The model takes an image, extracts a set of features from it and finally classifies it. We omit the region generation stage described in the original Faster R-CNN (Ren et al., 2015) since the location of the target object is known in our problem setup. Here, N denotes the number of possible object classes.

Lastly, the output of the ROI pooling is fed into three fully connected layers, and finally in a softmax layer, allowing the model to classify the target object. For more details about the sizes of each layer, refer to Fig. 2.2.

Through its use of state-of-the-art CNN for feature generation and ROI pooling for reducing complexity, Faster R-CNN manages to integrate both power and speed in tackling object recognition. An interesting artefact of the CNN processing is the integration of contextual information in the feature set. In our further analysis, we will explore how that use of contextual information affects the performance of the model on various image datasets.

## 2.2 DenseNet

Arguably, one of the most recent big advances in the field of object recognition was made by the paper given Best Paper Award in the 2017 Conference on Computer Vision and Pattern Recognition: "Densely Connected Convolutional Networks" (Huang et al., 2017). There, Huang et al. present their novel neural network architecture titled Dense Convolutional Network (DenseNet). DenseNet can be seen as a logical extension of ResNet (He et al., 2016), another very influential deep learning architecture. Both models make use of the idea that deeper networks and more features means more predictive power but DenseNet organizes its features in a more effective way, as we are going to see in this section.

One of the biggest improvements that DenseNet introduced in the field is the concept of *dense blocks* (Fig. 2.3). Dense blocks represent series of a batch norm (Santurkar et al., 2018), a ReLU activation, and a $3 \times 3$ convolution layers. Historically, layers in neural networks were almost always connected in a feedforward fashion without any jumping connections. However, in dense blocks, the layers are fully connected, meaning that every layer uses the outputs of the previous layers to compute a feature set. This strategy of reusing feature sets has showed a lot of promising benefits such as overall reduction of the number of parameters, strengthened feature propagation, and resolving problems such as the vanishing gradient problem (Hochreiter, 1998) and overall making it easier to train the network (Huang et al., 2017).

A crucial component of the dense block is its growth rate $k$. The growth rate indicates how many new feature sets are created after each layer. Since dense blocks effectively concatenate all of the generated feature sets, and no feature is wasted, the output of a dense block is its input combined with all of the generated features (Fig. 2.3). Huang et al. recommend using a growth rate of $k = 32$.

However, if the number of feature sets increase by 32 after every layer, how does DenseNet maintain a low number of parameters? To achieve this, the authors employ two strategies: the use of

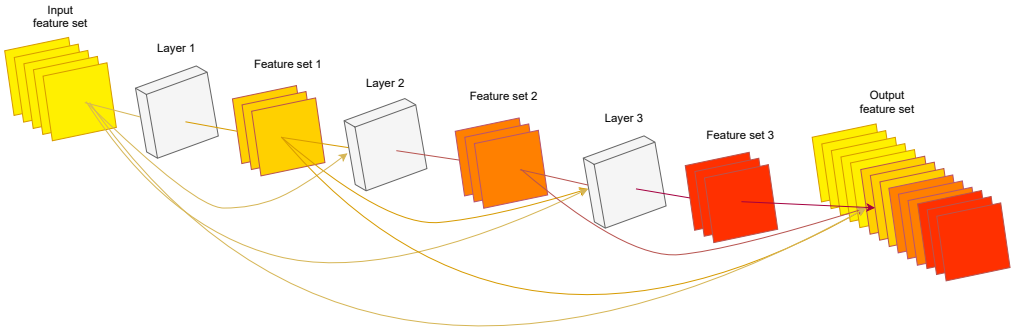



**Figure 2.3: A diagram of a dense block.** This figure presents a 3 layer dense block with a growth rate of $k = 3$. Each layer typically consists of a batch normalization, a ReLU activation, and a $3 \times 3$ convolution. The layers use all previous feature sets to produce a new one. The output of the denseblock is the concatenation of all feature sets.

narrow layers and bottlenecks. DenseNet exhibits lower need for wide layers due to its densely connected blocks and the reuse of features. Thus, before every dense block, an average pool with stride 2 is used to sample down the number of parameters in the feature sets. This reduces the size of the feature sets and allows the model to be more efficient. The second optimization involves the use of $1 \times 1$ convolution bottlenecks after every dense block which reduce the number of feature sets. The authors also perform compression to further improve the performance of the model.

Putting all of this together, the entire structure of DenseNet is shown in Fig. 2.4. The model uses a cropped image of the target object; it processes initial features with one convolutional and one max pool layer, after which it uses 4 dense blocks with their corresponding $1 \times 1$ convolutional bottlenecks and average pooling layers; and the model makes a final prediction in its fully connected layer with softmax activation. There are different versions of DenseNet depending on the total number of parameterized layers that they use. In our further analysis, we used DenseNet-169 which contains 12 convolutional layers in its first dense block, 24 in its second, and 64 in its third and fourth dense blocks (plus 4 other convolutional layers and one fully connected one, for a total of 169).

By integrating dense blocks, convolutional bottlenecks, and multiple different pooling layers, DenseNet managed to improve state-of-the-art results on ImageNet, CIFAR-10, CIFAR-100, and

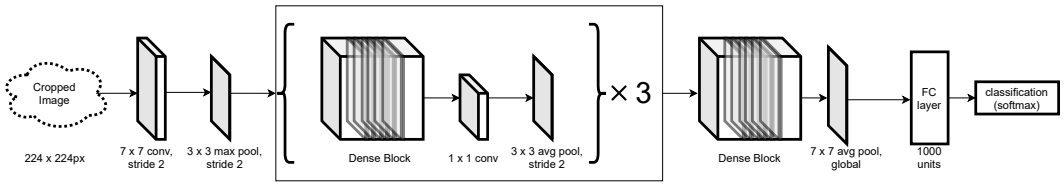



**Figure 2.4: A diagram of DenseNet that we used.** The model takes a cropped image of the target object resized to $224 \times 224$. Processing begins by extracting some features from the image via a set of convolutional layers and a max pool layer. After that the feature set goes through series of a dense block, a convolutional layer, and an average pool layer. After the last dense block, the feature map goes trough a global average pool layer. Finally, the model ends with a fully-connected layer and uses a softmax activation to classify the image. Here, N denotes the number of possible object classes.

SVHN, whilst requiring less computational power. An interesting artefact of the model is that, it does not use any contextual information, since it takes a cropped image of the target object. In our further analysis, we will aim to explore how that design choice affects the performance of the algorithm on in and out-of-context image recognition and how we can improve it.

## 2.3 CATNet

One of the recent models designed specifically for human-like object recognition is the Context-aware Two-stream Attention Network (CATNet) (Zhang et al., 2020). CATNet has shown promising results on object recognition tasks, but more importantly, it has also exhibited more human-like performance on these tasks (i.e. being able to recognize objects in conditions that are easy for people, and having difficulties recognizing the object when people also fail to do so). Unlike other object recognition algorithms, CATNet is able to implicitly incorporate contextual information in producing label predictions. Similar to how the human-eye processes information, CATNet treats the foveal (object) and peripheral (contextual) information separately, here labeled $I^o$ and $I^c$, respectively, and then aggregates the information to produce a class label.

Here, we describe CATNet's structure, consisting of three regions (also depicted in Fig. 2.5):

1. CNN Feature Extraction: This is similar to Feature extraction in Faster R-CNN (see Sec.

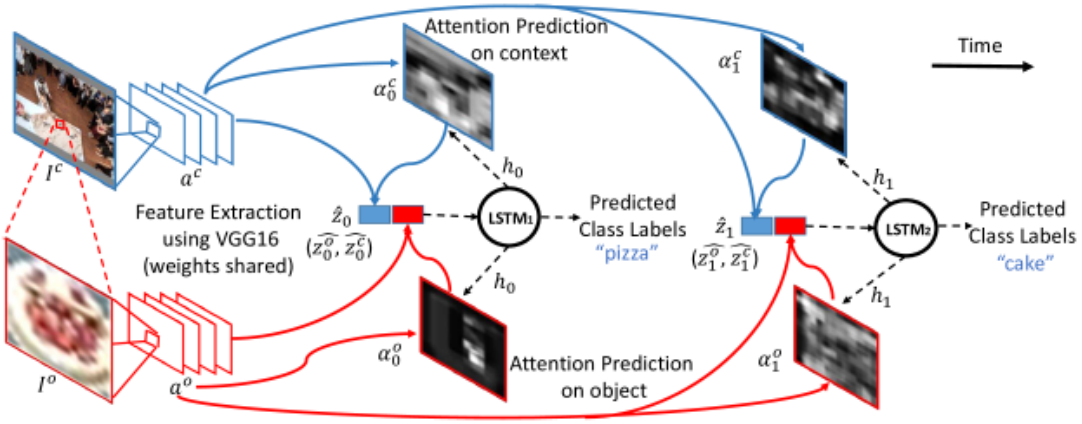



**Figure 2.5: A diagram of CATNet.** This diagram depicts CATNet's three regions: feature extraction, attention modulation, and recurrent memory. The first two regions process the object and context streams separately, whereas the third region produces a class label from their concatenation. These regions are repeated for a set number of time steps, the first two of which are illustrated here. Image taken with permission from Zhang et al., 2020.

2.1). The difference is that CATNet performs feature extraction for the object stream and context stream in parallel to produce an object and a context feature map.

2. Attention Modulation: This region mimics human cognitive attention by using a soft-attention mechanism (Ba et al., 2014) to enhance the important parts of the object and context information and fade the less important parts. This is done at each time step for the context and object streams in parallel and the outputs are "the context gist" and "the object gist."

3. Recurrent Memory: This final region uses a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) network to predict a class label for the target object at each time step from the attention region results.

Similarly to Faster R-CNN, CATNet also uses a feed-forward CNN such as VGG16 (Simonyan and Zisserman, 2014) pre-trained on ImageNet (Deng et al., 2009) to extract object and context feature maps. The context feature map, $a^c$, consists of feature vectors $a_i^c$ of dimension $D$. These

feature vectors represent the context input image $I^c$ at the location $i = 1, ..., W \times H$, where $W$ and $H$ are the width and height of the feature map, respectively.

$$a^c = \{a_1^c, ..., a_{W \times H}^c\}, a_i^c \in \mathbf{R}^D$$

The object feature map $a^o$ is of the same form but instead takes the input $I^o$.

The next region is the attention modulation, which specifically focuses on the important parts of the object and context information. This is done for both streams at each time step, ti. Here, we will explain only how the *context gist* is calculated from the context feature map, but the *object gist* follows the same steps.

A soft-attention module (Ba et al., 2014) is used to compute a scalar $a_{\text{ti}}^c$ for each location in $a^c$, that indicates the relative importance of that location in capturing the context gist. All attended regions are not necessarily useful for context reasoning, so the soft-attention module is also used to predict a gating vector that represents the relative importance of each contextual observation (Xu et al., 2015). The model uses the context attention map combined with the previous hidden state of a recurrent neural network, labeled $\alpha_{\text{ti}}^c$, and the gating vector, labeled $\text{B}_{\text{ti}}^c$ to calculate $\hat{z}_t^c$, the context gist, as follows:

$$\hat{z}_t^c = \Sigma_{i=1}^{W \times H} \text{B}_{\text{ti}}^c \alpha_{\text{ti}}^c a_i^c$$

$\hat{z}_t^o$ is calculated in a similar manner.

The final region, the recurrent memory, combines both streams and uses an LSTM network (Hochreiter and Schmidhuber, 1997) to predict the class label $y_t$ for target objects at each time step (Zaremba et al., 2014). The LSTM uses the overall gist vector, $\hat{z}_t$, which is a concatenation of the context gist and object gist, $\hat{z}_t^c$ and $\hat{z}_t^o$. The label $y_t$ is predicted by computing a classification vector given the hidden state of the LSTM network. The most probable class from the classification vector

is used as the predicted label $y_t$ at the current time step.

One of the most important novel contributions of CATNet includes the utilization an attention modulation, combined with a recurrent memory used for processing both object and context information. By attenuating to contextual cues, CATNet has shown promising results in the field of out-of-context object recognition. It has exhibited high accuracy but even more importantly more human-like recognition when it comes to which images are easier or harder to recognize. In our further analysis, we will build off CATNet's attention module and explore how it can be improved.

# 3

# Context-aware Recognition Transformer Network

Inspired by the robustness of human object recognition and by previous work in the field of computer vision, we developed the Context-aware Recognition Transformer Network (CRTNet). CRTNet combines advantages from Faster R-CNN (integrated contextual information) (Ren et al., 2015), DenseNet (powerful feature map) (Huang et al., 2017), and CATNet (the use of attention)

(Zhang et al., 2020) to recognize objects in a more accurate, more robust, and more human-like way. In this chapter, we aim to describe the model and offer the motivation behind our design choices. We will start by giving an overview of how CRTNet works and then we will discuss each part of the network in depth.

## 3.1 Overview and Motivation

Contextual information matters. Multiple studies have shown that context can be both beneficial and detrimental to the performance of neural networks (Rosenfeld et al., 2018) (Singh et al., 2020) (Zhang et al., 2020). However, this fact is not utilized in some state-of-the-art networks like DenseNet which processes only the target object while ignoring all contextual information. We designed CRTNet that utilizes DenseNet's powerful feature extraction and improves upon it by integrating contextual information in the model. To achieve that, we decided to use a two-stream network that processes contextual information in one stream and the isolated target object in the other stream (see Fig. 3.1).

Therefore, our model needs to take two images: the target object image with no contextual information $I_t$, and the original image containing all of the contextual information $I_c$. Note that $I_t$ is the original image cropped around the bounding box of the target object. Similarly to DenseNet, both $I_t$ and $I_c$ are resized to $224 \times 224$ before they are passed to the network. After the images are passed to CRTNet, they are independently and simultaneously processed by DenseNet-like feature extractor networks. These networks produce feature maps $a_c$ and $a_t$ that are further processed in the two steams of the model.

The context stream of CRTNet uses both feature maps $a_c$ and $a_t$, and it passes them through a transformer decoder (Vaswani et al., 2017). The transformer decoder consists of 6 layers and follows the original structure described by Vaswani et al., 2017. Every layer of the transformer decoder at-

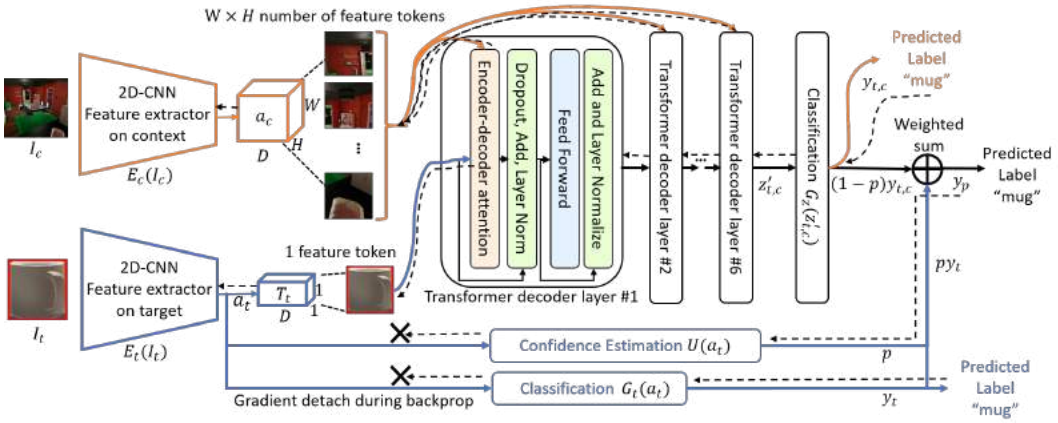



**Figure 3.1: A diagram of CRTNet.** Image created jointly with Mengmi Zhang. The diagram depicts the architecture of CRTNet. The network consists of 3 main modules: feature extraction, context stream that integrates context and target information via a transformer decoder, and the target stream which consists of a confidence estimation and classification. CRTNet takes the cropped target object $I_t$ and the entire context image $I_c$ as inputs and extracts their respective features. These feature maps are then tokenized and the information of the two streams is integrated over multiple transformer decoding layers. CRTNet also estimates a confidence score of recognizing the target object based on object features alone, which is used to modulate the contributions of $y_t$ and $y_{t,c}$ to the final prediction $y_p$. The dashed lines in a backward direction denote gradient flows during backpropagation. The two black crosses denote where the gradient updates stop.

tenuates to parts of the context while also processing the target object at the same time. The output of the last layer is put through a classification layer with a softmax activation to give us a prediction vector.

The target stream only uses the target image feature maps. Without further processing, the stream passes $a_t$ through a classification layer which produces another prediction vector. The model estimates how confident it is that prediction, after which it combines the prediction vectors obtained from both streams in a weighted sum and makes a final decision about the object's identity.

## 3.2 Feature Extraction

As discussed above, CRTNet takes $I_c$ and $I_t$ and passes them through 2D-CNN feature extractors. The feature extractors are implemented by using a DenseNet-like architecture pre-trained on Ima-

geNet (Deng et al., 2009). More concretely, we use a modification of DenseNet-169 containing only the convolutional layers without the last fully connected layer.

The outputs of the feature extractors are $a_c$ and $a_t$. Both $a_c$ and $a_t$ are of size $D \times W \times H$, where $D$ is the final number of channels, $W$ is the width, and $H$ is the height of the feature maps. Consistent with the original implementation of DenseNet-169, our final feature maps are of size $1664 \times 7 \times 7$. Note that in both $a_c$ and $a_t$, the spatial organization of the features is preserved, so the following layers can make use of the contextual information as it appears in $I_c$. Furthermore, we do not enforce parameter sharing, so when we finetune the network, the weights in the two feature extractors will be different. This is consistent with our assumption that contextual information might contribute in different ways to recognizing the object than direct information about the target object.

## 3.3 Tokenization and Positional Encoding

After obtaining the feature maps from the feature extractors, CRTNet needs to tokenize them before passing them to the transformer decoder. To tokenize $a_c$, we follow a simple procedure described by (Dosovitskiy et al., 2020). We create $W \times H$ tokens, where token $i$ contains all of the channels at location $i$ from the feature map. Therefore, all of our tokens are of size $D$. Note that since $a_c$ preserves the spatial organization of $I_c$, each token will represent exactly one location of the image.

In order to tokenize $a_t$, we repeat the same procedure but after obtaining all of the $W \times H$ tokens, we aggregate them through average pooling. We end up with one token $T_t$ of size $D$.

$$T_t = \frac{1}{W \cdot H} \sum_{i=1,\ldots,W \cdot H} \mathbf{a_t^i} \tag{3.1}$$

To identify the location of each token in the original image, CRTNet learns a positional embedding $a_p^i \in \mathbb{R}^D$ for each location $i$ of the image (see Fig. 3.2). For the target token $T_t$, we use the

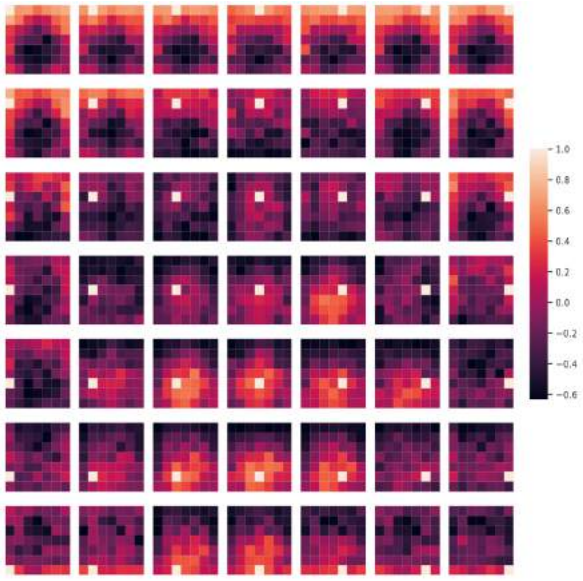



**Figure 3.2: CRTNet learns reasonable positional embeddings for feature tokens.** The figure shows the similarity of position embeddings of CRTNet. Each tile shows the cosine similarity between the position embeddings of the patch with the indicated row and column and the position embeddings of all other patches. We can see that in general, the positional embeddings are more similar if they are closer to each other. See the color bar on the right for cosine similarity values. Figure generated by Philipp Bomatter.

positional embedding corresponding to the location, within which the bounding box midpoint is contained. We denote the positionally-encoded context and target tokens by $z_c$ and $z_t$ respectively.

$$z_c = a_c + a_p \qquad a_p \in \mathbb{R}^{(W \cdot H) \times D} \tag{3.2}$$

$$z_t = T_t + a_p^l \qquad a_p^l \in \mathbb{R}^D \tag{3.3}$$

The new positionally-encoded tokens are then forwarded to the transformer decoder.

## 3.4 Transformer Decoder

The transformer decoder that we used followed very closely the original architecture proposed by Vaswani et al., 2017. It consists of 6 layers, where each layer takes as input all of the context tokens:

$z_c$ and the output of the previous layer (or $z_t$, if it is the first layer). The transformer decoder layers use $z_c$ to compute keys and values, and $z_t$ or the output of the previous layer to generate queries in the transformer encoder-decoder multi-head attention layer.

The structure of every layers is as follows. It starts with a typical encoder-decoder attention layer (EDA), the output of which is passed through a dropout layer (DROP) and a layer normalization (LN). This is followed by a feed-forward multi-layer perceptron network (MLP), the output of which is also passed through a dropout and layer normalization. The encoder-decoder attention layer allows the network to attenuate to different contextual information within the image, while the feed-forward network allows it to integrate that information. The dropout and layer normalization help to reduce overfitting, smooth gradients, and make training easier. The multi-layer perceptron network consist of two layers with a ReLU non-linearity. All of this is summarized in the following equations:

$$z_{t,c} = \mathrm{LN}(\mathrm{DROP}(\mathrm{EDA}(z_t, z_c)) + z_t) \tag{3.4}$$

$$z'_{t,c} = \mathrm{LN}(\mathrm{DROP}(\mathrm{MLP}(z_{t,c})) + z_{t,c}) \tag{3.5}$$

Each of the 6 layers performs the calculations above and passes the result ($z'_{t,c}$) to the next layer (and becomes the new $z_t$). We use 8-headed selective attention as suggested in the original transformer (Vaswani et al., 2017) to ensure that the model will concentrate on different contextual information. We provide an example visualization of the transformer attention in Fig. 3.3.

## 3.5 Label Prediction

The context stream of CRTNet ends with classification layer ($G_z$) which consists of a fully-connected layer and a softmax layer. The classification layer takes as an input the output of the last transformer


**Figure 3.3: Visualization of attention maps over all transformer decoding layers.** We show the attention map averaged over all attention heads within the same layer. The original image is shown in the first column (the top two rows show examples from the COCO-Stuff dataset (Caesar et al., 2018) and the bottom two rows show examples from our VirtualHome dataset. We can see that the transformer decoder ultimately pays attention to important parts of the image (e.g. the toilet in the bathroom contains more information about the room than the wall). Figure generated by Philipp Bomatter.

decoder layer $z'_{t,c}$ and predicts a label $y_{t,c}$:

$$y_{t,c} = G_z(z'_{t,c}) \tag{3.6}$$

Similarly, the target stream uses the target feature map $a_t$ to make a prediction $y_t$ with a classification layer $G_t$ that has the same structure as $G_z$, but potentially different weights. However, the target stream also calculates a confidence estimation $p$ for $y_t$ through a feed-forward multi-layer perceptron confidence estimator $U$. Additionally, a sigmoid function is used to normalize the confidence score to $[0, 1]$.

$$y_t = G_t(a_t) \tag{3.7}$$

$$p = \frac{1}{1 + e^{-U(a_t)}} \tag{3.8}$$

The main purpose of $p$ is to inform the network to what extend it should use the prediction from the target stream $y_t$ versus the prediction from the context stream $y_{t,c}$. To express that, we calculate a final prediction $y_p$ that is a confidence-weighted average of $y_{t,c}$ and $y_t$:

$$y_p = py_t + (1 - p)y_{t,c} \tag{3.9}$$

Here it is worth noting a few consequences of the confidence estimation. If $p = 1$, then CRTNet would have been essentially the same as DenseNet. While this can be very beneficial when the network clearly recognizes the object, it can also be detrimental if the context is useful. On the other hand, if $p = 0$, then we rely on the entire image including context. While this allows us to use all of the available information, it can also be detrimental if the model relies too much on faulty context. Therefore, we introduced a way to learn to assign $p$, so that CRTNet relies on contextual information only if it cannot recognize the object on its own. This way, we can both benefit from

contextual information when it is useful and learn to ignore it when we do not need it.

## 3.6 Training

Finally, we will discuss how the network was trained. We used three different cross-entropy losses: with respect to $y_t$, $y_p$, and $y_{t,c}$ (see Fig. 3.1).

1. We trained $G_t$ with respect to $y_t$. This allows $G_t$ to concentrate on improving its classification of $a_t$ and does not add other unknowns that might have been introduced if we trained on $y_p$.

2. We trained $U$ with respect to $y_p$. Similarly, as above, the confidence estimator affects only $y_p$, and thus it makes sense for it to be trained only with respect to it. Intuitively, this allows $U$ to increase $p$ if $a_t$ on its own was responsible for that prediction.

3. We trained the rest of the network with respect to $y_{t,c}$.

Note that the feature extractors are also trained with respect to $y_{t,c}$, even though the target feature extractor influences both $U$ and $G_t$ (in Fig. 3.1 this gradient detachment is denoted by the two black crosses). We made this decision because we did not want the accuracy of the transformer decoder to be affected by the target stream and $y_t$. Further testing with and without detachment of the gradient in the target stream proved that CRTNet indeed performs better when the gradient is detached.

# 4

# Datasets

The main goal of this project is to study how context influences object recognition. To do that, we need image datasets that allow us to change only the features of the context that we want to study, and control for all other unintended effects. Here we describe two datasets that we used for our further analysis: (1) the VirtualHome dataset, which is our original dataset created via manipulations in a 3D synthetic environment, and (2) the Cut-and-paste dataset, which is a dataset of natural images created by Zhang et al., 2020 and designed for studying the effects of context.

## 4.1 VirtualHome Dataset

To create our original dataset, we decided to use VirtualHome – a platform that allows for simulation of complex household activities and control over a variety of properties of everyday objects in a 3D environment (Puig et al., 2018) (Liao et al., 2019) (Puig et al., 2020). VirtualHome strikes a perfect balance between being a powerful platform that allows for complex object manipulations and being easily controlled through a Python API.

VirtualHome supports 225 different objects and offers 7 different virtual apartments each consisting of 5 rooms that include furnished bedrooms, kitchens, study rooms, living rooms and bathrooms. VirtualHome allows users to programmatically add, remove and modify objects through *the environment graph*. The environment graph is a way to describe the state of the apartments. It represents objects as nodes and spatial relationships as edges. The graph can be used to query information about the apartments but also change their states. We used the environment graph to modify properties of the objects in the apartments, and then we took series of pictures from different angles of the modified objects to be used in our datasets. For the rest of this section, we proceed with description of the methods used to create the six context conditions that we used: in-context, gravity violation, co-occurrence statistics violation, size violation, a combination of co-occurrence statistics and gravity violation, and a no context condition (see Fig. 4.1).

### 4.1.1 In-context Objects

First, we needed to generate a dataset that would be used as a control condition when studying context. To do that, we generated images of objects in their natural context. Note that defining "natural context" is not a trivial task since there are often multiple ways to do this. To determine what "natural context" means in this scenario, everyone working on this project filled a matrix indicating which locations are natural for which objects. The average of everyone's responses can be seen in Table A.1

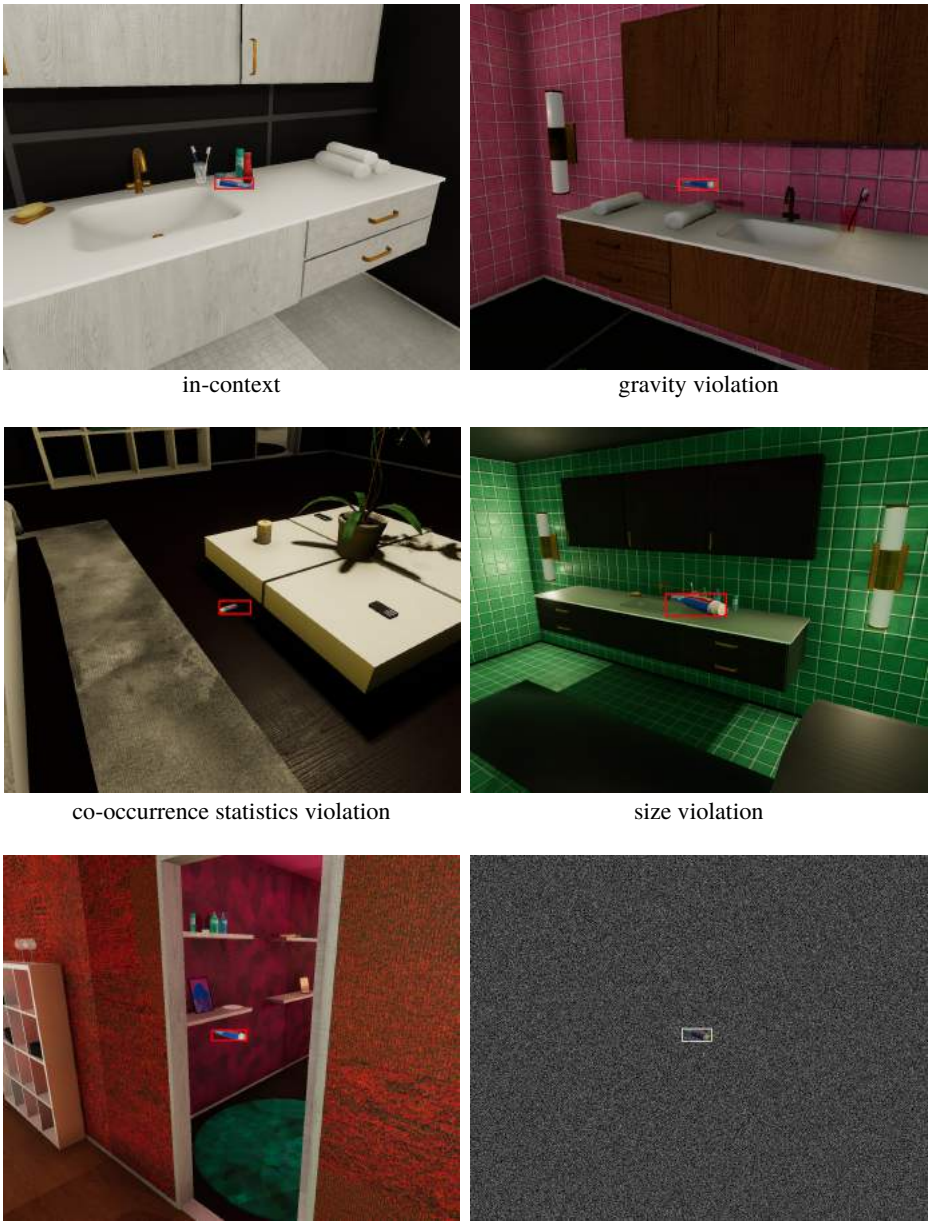



**Figure 4.1: Examples of a toothpaste placed in all six contextual conditions.**

and Table A.2.

To generate the images, we placed every object in each of the locations that are considered natural for it (e.g. a remote control on a TV stand or a toothpaste on the bathroom counter). For every object placement, we took pictures of the object from a maximum of 9 fixed view angles. The azimuth angles ranged from 0 to 320 degrees in steps of 40 degrees, fixed elevation angle of 19.5 degrees and radius of 1.5 meters; in some special cases, the elevation angle was set to −19.5 degrees to prevent the camera from penetrating the ceiling.

After filtering, the code produced 2,309 unique images. Refer to Fig. 4.2 for example images in this condition.

Here and in all other experimental conditions, we used the following 36 object categories: apple, bar soap, book, candle, cellphone, cereal, Chinese food, chocolate syrup, coffee maker, condiment bottle, condiment shaker, cupcake, cutlery knife, cutlets, dish bowl, dishwashing liquid, keyboard, lime, microwave, milkshake, mouse, mug, peach, pie, pillow, plate, plum, pound cake, pudding, remote control, slippers, toothbrush, toothpaste, towel, washing sponge, wine glass. For example images of every category, refer to Fig. 4.2, Fig. 4.3, Fig. 4.4, Fig. 4.6, Fig. 4.7, and Fig. 4.8.

Generating pictures of in-context objects served a dual purpose. On one hand, we were able to obtain a neutral dataset that we could use for a control condition. On the other hand, we were able to obtain many different natural positions for the objects which were used as a base for generating the contextual irregularities in our other conditions (with the exception of the co-occurrence violation condition). Note that this allows us to change the context in the images in a controlled manner – one element at a time.

### 4.1.2 Gravity Violation

*Intuitive physics* is the hypothesis that humans understand Newtonian physics on an intuitive level and are able to predict a wide variety of physical events (McCloskey, 1983). There is a growing

dishwashing liquid

mouse

peach

pillow

plate

towel

**Figure 4.2: Examples of in-context images.**

amount of literature suggesting that intuitive physics plays a crucial role in a wide variety of tasks such as mass approximation, estimating the projection of a falling object, and is even inversely correlated with our ability to understand other people's intentions (Baron-Cohen et al., 2001) (Kubricht et al., 2017). There is even some effort to create computational models of intuitive physics and introduce it in neural networks (Battaglia et al., 2012) (Agrawal et al., 2016).

Following the discussion on intuitive physics in the field, we wanted to see if violating a law of physics such as gravity will have any effects on object recognition in both humans and machines. To do that, we lifted the in-context objects up by 1 meter and proceeded to take multiple pictures of them from different angles in a similar manner as described in the previous subsection.

After filtering, the code produced 1,453 unique images. Refer to Fig. 4.3 for example images in this condition.

### 4.1.3 Co-occurrence Statistics Violation

Objects do not exist in isolation – they co-occur with backgrounds, other objects, and various scene properties. You will probably never see an oven in the ocean or coral reefs in your kitchen but swapping oven and coral reef yields much more likely combinations. It has been shown that humans recognize an object faster and more accurately if the object is presented in a typical context (e.g. an oven in the kitchen) (Bar and Ullman, 1996) (Bar, 2004). Similar and even more drastic results have been shown for various object recognition models that are often much more sensitive to contextual changes than humans (Rosenfeld et al., 2018) (Zhang et al., 2020).

We wanted to study how exactly these contextual changes affect the performance of humans and state-of-the-art neural networks on object recognition tasks. To do that, we placed objects in atypical locations for them as described in Section 4.1.1 and in accordance to Table A.1 and Table A.2 (e.g. a remote control in the sink). As before, we took pictures from different angles around the object.

After filtering, the code produced 2,363 unique images (see Fig. 4.4 for example images).

coffee maker

cutlets

keyboard

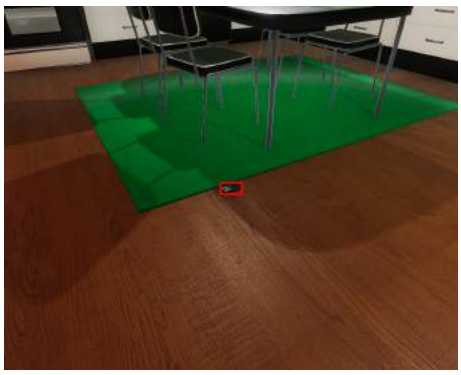
toothbrush

toothpaste

washing sponge

**Figure 4.3: Examples of gravity violated images.**

book



cellphone

cutlery knife

lime

microwave

pudding

**Figure 4.4: Examples of co-occurrence statistics violated images.**

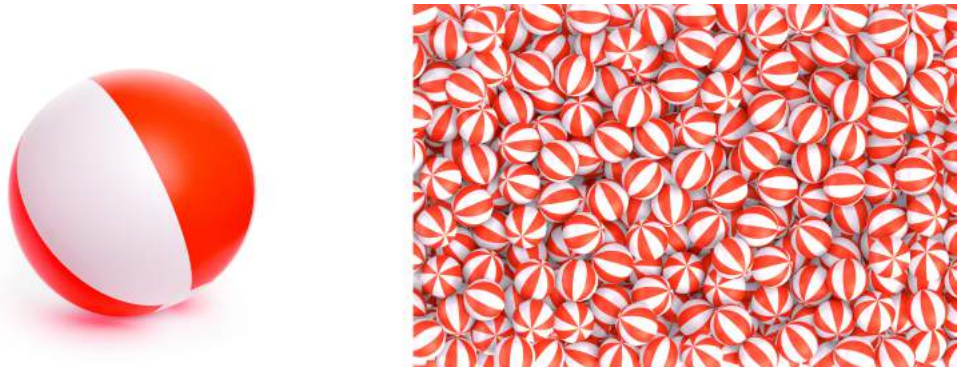


**Figure 4.5: Size matters for object recognition.** Even though the objects in these images look exactly the same, most people will recognize the object in the first picture as a beach ball and the objects in the second picture as candy. Pictures obtained from Shutterstock under Standard License. Authors: Feng Yu and Oguzhan Ayvazoglu.

### 4.1.4 Size Regularities Violation

It is safe to say that size matters to some extent when it comes to object recognition. For example, in Fig. 4.5, the small circular object on the table is most likely candy whereas the big identical one is most likely a beach ball. Intuitively, smaller objects should typically be harder to recognize than bigger objects due to the fact that visually small objects hold less information. This intuition is confirmed by various studies (Tong et al., 2020) (Zhang et al., 2020). However, still little is known about changing the *expected* size of an object (i.e. the relative size to other objects in the scene) and how that would affect object recognition accuracy.

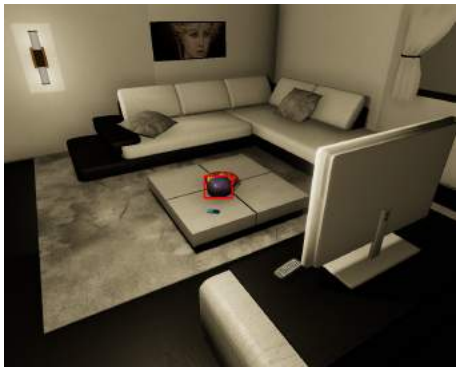
To get a better insight of how size affects object recognition, we modified the in-context objects from Section (4.1.1) by increasing their size. We increased their size by a factor of 2, 3, and 4 in order to study the effects of size as the multiplier increases.

After filtering, the code produced 5,858 unique images. Refer to Fig. 4.6 for example images in this condition.


chocolate syrup

condiment bottle

pie



plum

pound cake

remote control

**Figure 4.6: Examples of size regularities violated images.**

### 4.1.5 Co-occurrence and Gravity Violation

We also decided to generate images in a condition that simulates both co-occurrence statistics violation and gravity violation in order to study how the negative effects of two unnatural context conditions add-up. The code produced 910 images for this condition (see Fig. 4.7 for example images).

### 4.1.6 No Context Objects

The no context condition presents the objects on a gray "salt and pepper" background and allows us to obtain an estimate of how humans and machines perform when there is no contextual information in the image. There are 2,309 images for this condition (see Fig. 4.8).

### 4.1.7 Filtering out Bad Images

One of the hardest parts about the data generation was filtering out the "bad images". Due to the exhaustive nature of the data generation process, many low quality images were generated and fell in one of the following scenarios:

1. The camera was outside of the room or the apartment.
2. The virtual environment was too dark due to bad lighting caused by bugs in VirtualHome.
3. The target object was completely occluded by a different object.
4. The target object was partially occluded by a different object.
5. The target object collided with another object (in the size and gravity conditions).

We were able to programatically filter out images that fell in scenarios 1-3. However, scenarios 4-5 were much more challenging due to difficulty defining how much occlusion is too much and

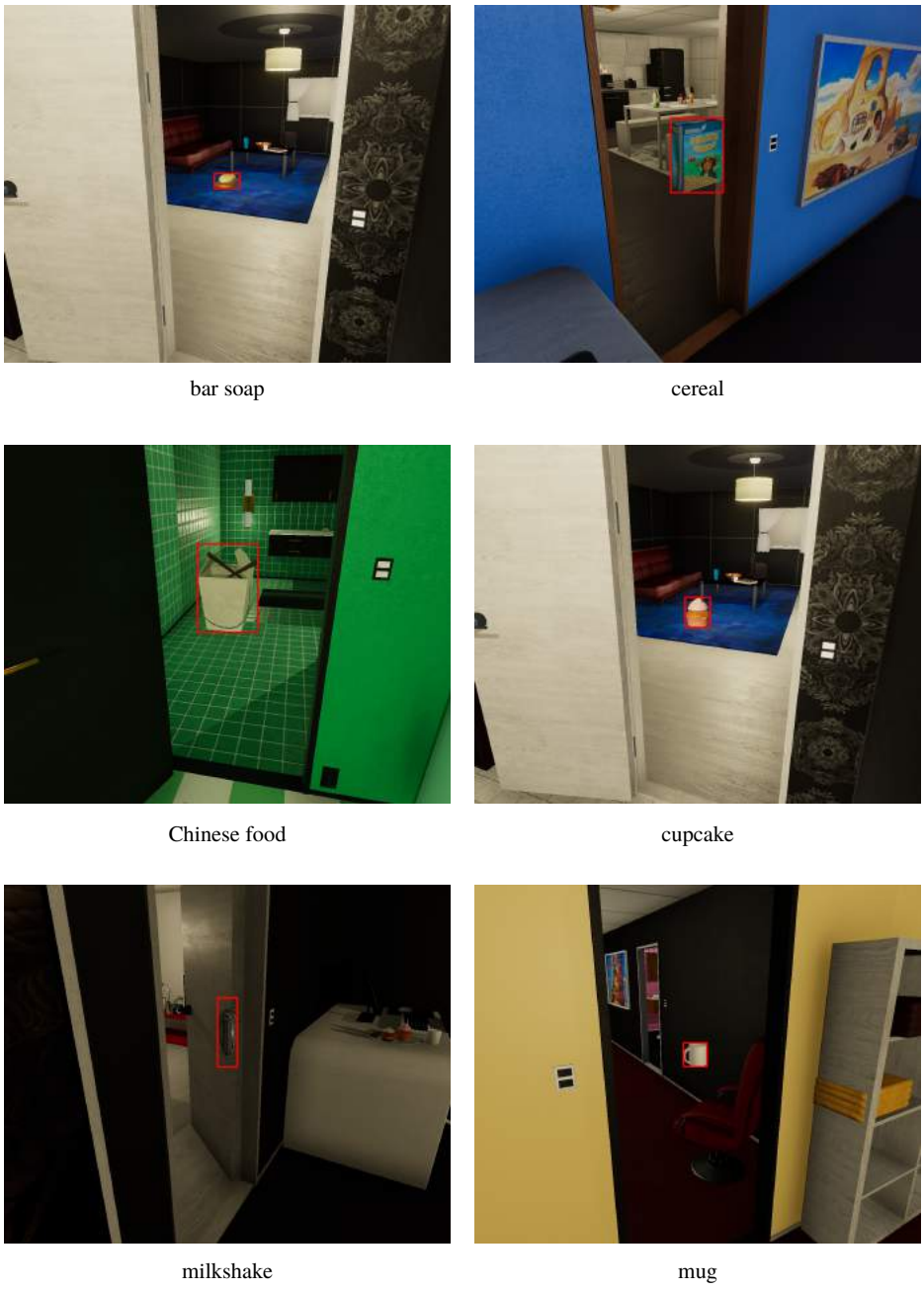
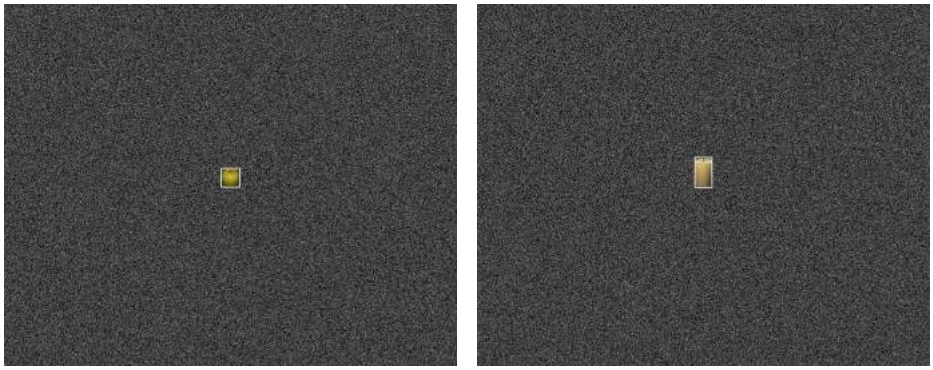
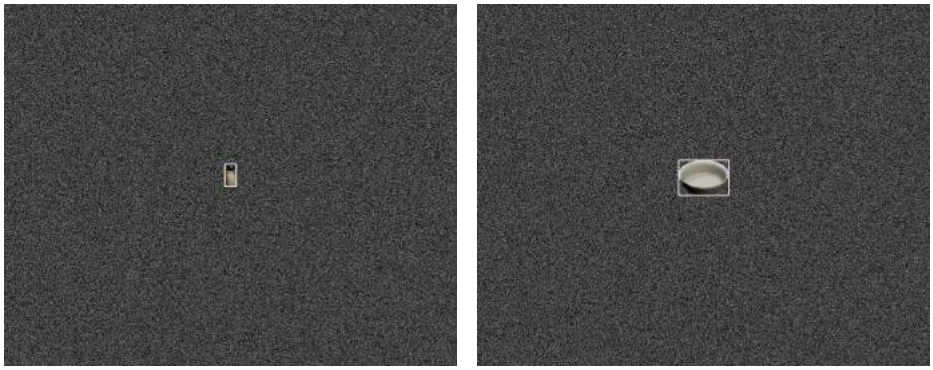


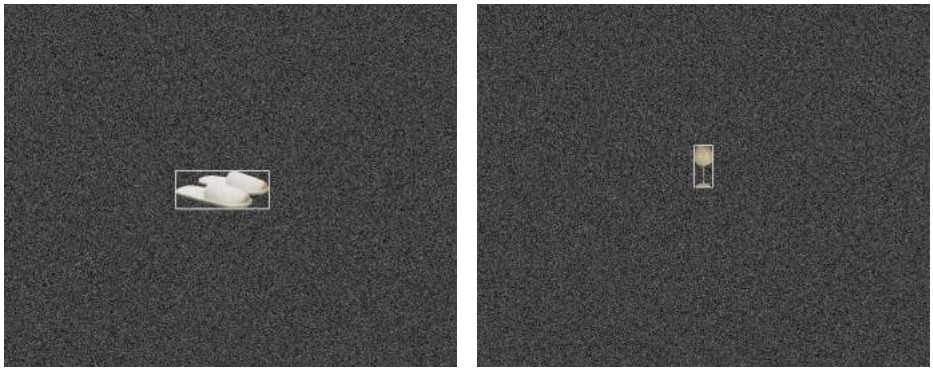

**Figure 4.7: Examples of co-occurrence statistics violation and gravity violation images.**



apple candle



condiment shaker dish bowl



slippers wine glass

**Figure 4.8: Examples of no context images.**

VirtualHome not supporting collision detection. Therefore, we needed to manually remove all images that were deemed bad which in some cases meant going through more than 10000 images.

To assist with the manual work, we designed a graphical user interface that allows the user to quickly go through many images and label them as either "good" or "bad". The graphical user interface logs all of the labeling decisions made by the user and it also supports functions to preview all images with certain label. Please find the complete code, written in Python 3, in Appendix B.

## 4.2 Cut-and-paste Dataset

In addition to our VirtualHome data, in our further analysis we will also use the cut-and-paste dataset created by Zhang et al., 2020. Zhang et al. modified a popular dataset of images placed in natural context – the MS COCO Dataset (Lin et al., 2014). The resultant dataset contained images grouped into 16 conditions – combinations of 4 object sizes and 4 context conditions: normal, minimal, congruent, and incongruent.

The *normal context* condition is just the original image without any further modifications. The *minimal context* condition includes only the target object and all pixels that happen to be in its bounding box while everything else is grayed-out (note that this is different than the no context condition that we used in the VirtualHome dataset). The congruent and incongruent conditions were created by cutting the target objects and placing them in the same location in different images (thus the name of the dataset). A *congruent context* is when the target object is placed in an image that contains an object with the same class label (see Fig. 4.9 c), whereas an *incongruent context* is when the target object is placed in an image that does not contain that object (see Fig. 4.9 d).

All images are also grouped based on the size of the target object in order to quantify how object size affects accuracy. There are four size bins based on the degrees of visual angle (dva): Size 1 dva [16-32 pixels], Size 2 dva [56-72 pixels], Size 4 dva [112-144 pixels], and Size 8 dva [224-288 pixels].

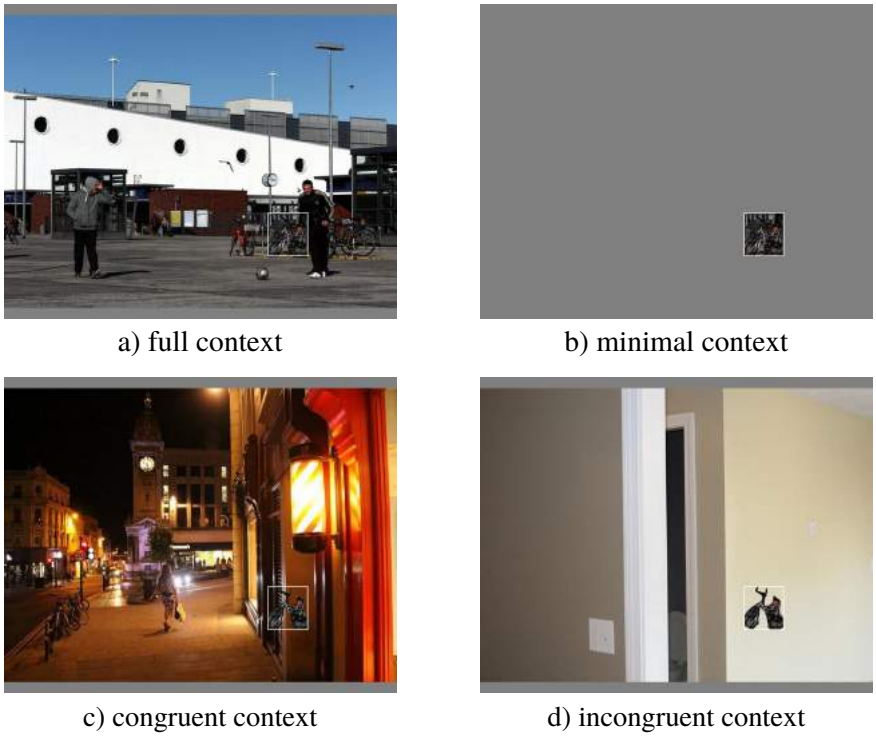



**Figure 4.9: Examples of the four contextual conditions in the cut-and-paste dataset.**

The final dataset contains 2,259 unique in-context images spanning 55 object categories which gives us more than 9,000 images across the four context condition described above.

In our further experiments, we use a subset of the cut-and-paste dataset containing 7,772 images. All images in our dataset participate with their 4 versions (i.e. a full context image, a minimal context images, a congruent image, and an incongruent image). This allows us to study context in a controlled manner by ensuring that all of the recognition effects we see are caused by changes in contextual information and not by chance.

# 5

# Human psychophysics experiments

A lot of prior studies exist on how humans recognize objects and how successful they are on various image datasets (Thorpe et al., 1996) (Grill-Spector and Malach, 2004) (DiCarlo et al., 2012) (Geirhos et al., 2017). However, we wanted to test humans on our new VirtualHome dataset for two main reasons (1) to assess human performance on the task of out-of-context object recognition specifically in the case of our dataset, and (2) to create a benchmark for our computational models and potentially to assess which models behave in a more human-like way.

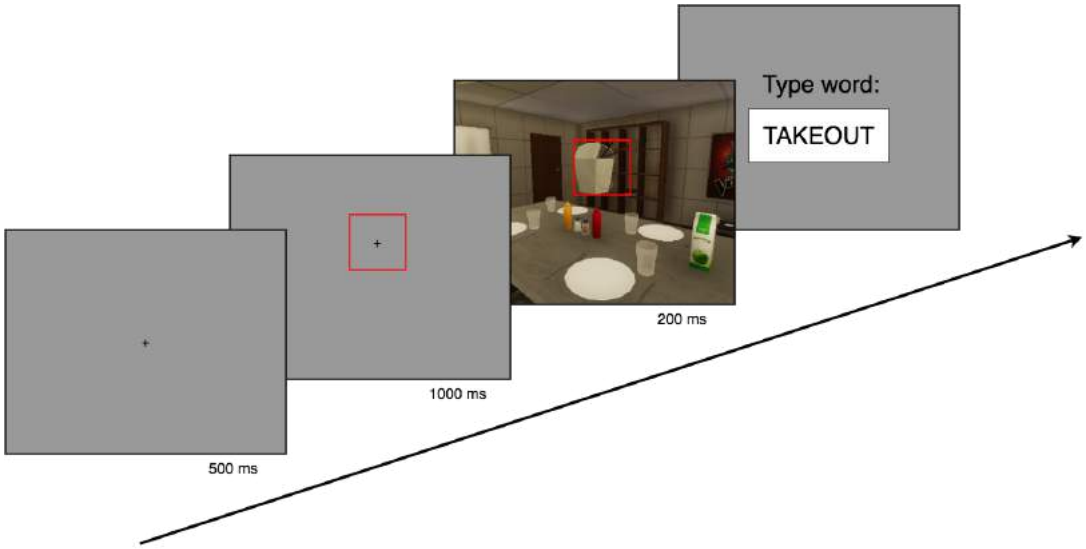



**Figure 5.1: Experiment Design.** Human participants were presented with series of trials, where each trial consisted of four stages. First, participants were presented with a fixation cross for 500 ms. Then, the location of the object of interest was revealed (as indicated by the red bounding box) for 1 second. After that, the entire image was flashed for 200 ms. Finally, the subjects were prompted to enter the name of the object that they saw.

We evaluated human recognition in the six contextual conditions described above: normal context, gravity violation, co-occurrence statistics violation, size regularities violation, gravity and co-occurrence statistics violation, and no context. We conducted psychophysics experiments on Amazon Mechanical Turk, an online crowdsourcing tool that is often used for data validation (Turk, 2012). We exposed the subjects to each image for 200 ms, after which we asked them to identify the target object encapsulated by a red bounding box. The full experimental setup can be seen in Fig. 5.1.

Four hundred participants took part in each experiment, for a total of around 67, 000 trials. To avoid any other confounding factors, we took the following precautions: (1) only one target object from each class category was selected; (2) each subject saw only one room in every apartment once; (3) the trial order was randomized.

To determine whether the subject gave the correct answer in each trial, we used a technique in-

troduced by previous studies (Zhang et al., 2020) to obtain a set of correct responses for each object class. We conducted an initial ground-truth experiment with images from our in-context condition where participants were given infinite viewing time. Answers in our further experiments were deemed correct if they matched any of the ground truth responses (e.g. other correct answers for the object presented in Fig. 5.1 include box, Chinese food, and container). This allowed us to account for synonyms without enforcing an *N*-way categorization commonly used by computational models and other human object recognition experiments. Finally, to avoid biases, we did not allow Amazon Mechanical Turk subjects that participated in the ground truth study to also participate in any of the other studies.

# 6

# Results

## 6.1 Human Recognition in VirtualHome

We performed the experiments described in Section 5. Figure 6.1 summarizes our findings. As expected, humans achieve very high accuracy on the task of object recognition in VirtualHome: 71% on average across all conditions. Here, it is worth noting that if the participants had infinite viewing time, we would expect them to have close to perfect recognition accuracy on the dataset. Therefore,

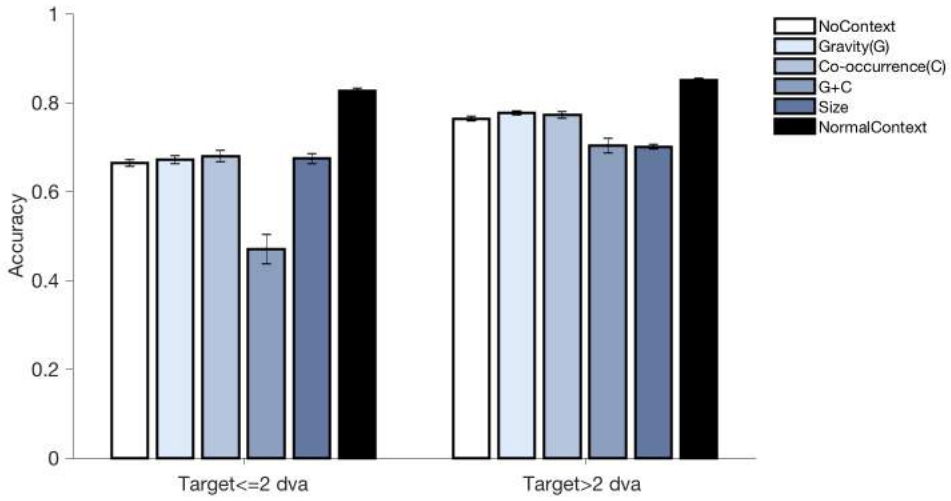



**Figure 6.1: Human Psychophysics Results on All Object Categories.** Results of the human psychophysics expriments (see Sec. 5) on all conditions from our VirtualHome dataset. We split the results in two bins based on the size of the target object (less and more than 2 degrees of visual angle). Here and in all future figures, the error bars represent standard error of the mean.

we hope that the effects of this study are mainly due to the different conditions that the objects are placed in.

We can notice the following facts about human recognition from our results:

1. **Recognizing smaller objects is harder than recognizing bigger objects.** Consistent with previous studies (Tong et al., 2020) (Zhang et al., 2020), we found that objects that are smaller than 2 degrees of visual angle are significantly harder to recognize (66% accuracy) than objects that are bigger than that (76% accuracy). This is also true for all computational models as we are going to see in the next subsections (see Figs 6.3, 6.5 and Table 6.2).

2. **Recognizing unnaturally big objects is harder than recognizing objects of a typical size.** While bigger objects are generally easier to recognize, artificially enlarged objects are harder to recognize. Figure 6.2 summarizes the results from the experiment on size regularities violation. We can see that accuracy drops as we increase the size of the object.

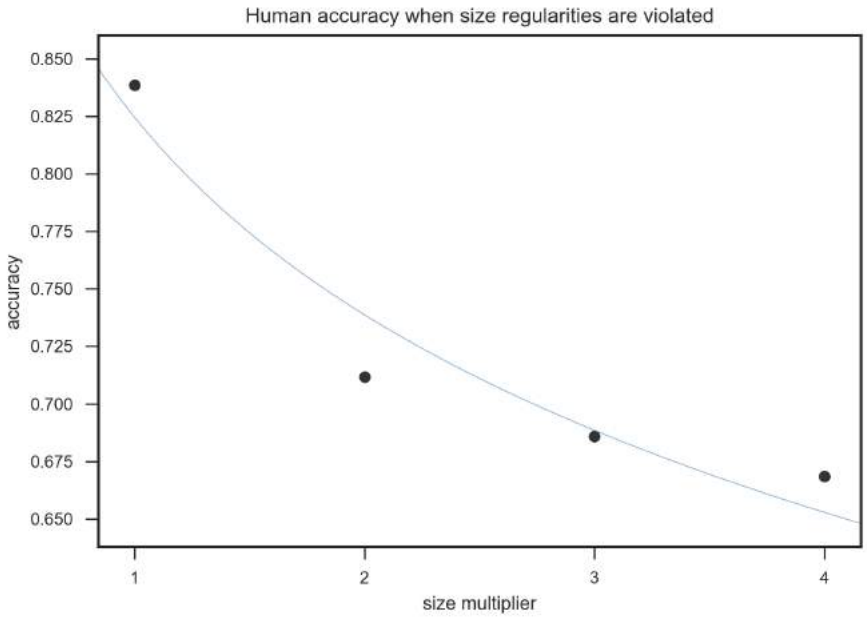



**Figure 6.2: Human Psychophysics Results on the Size Experiment.** Human recognition accuracy decreases as objects become artificially bigger. A log-linear regression line is fitted.

3. **Context matters less for big, easily recognizable object.** The difference in accuracy between normal context and the other conditions on average is 19.5% for small objects versus 10.7% for big objects. Intuitively, we can explain this difference by the fact that as object size increases, there is more visual information available, which means that the object will be more easily recognizable. Once the object is easier to recognize, context matters less, and thus the difference between normal context and the other conditions is smaller when the object is bigger.

4. **Context can help recognition.** In all of our results in-context objects are much easier to recognize than out-of-context objects, which suggests that normal context helps human object recognition. This is particularly apparent when we compare the no context condition

with normal context. We can also see this trend by the increase in accuracy from no context in the gravity and co-occurrence conditions. We hypothesize that that increase occurs due to the fact that even partially unusual context might help recognition (e.g. people can still infer that the target objects are household items by the fact that they see them in an apartment).

5. **Context can hurt recognition.** Even though context can be helpful, it can also hurt recognition. For both small and big objects, we can see that the gravity and co-occurrence violation condition has significantly lower accuracy than the no context condition. This indicates that if the context is too unusual, it can be misleading and it can result in lower accuracy.

## 6.2 Computational Model Recognition in VirtualHome

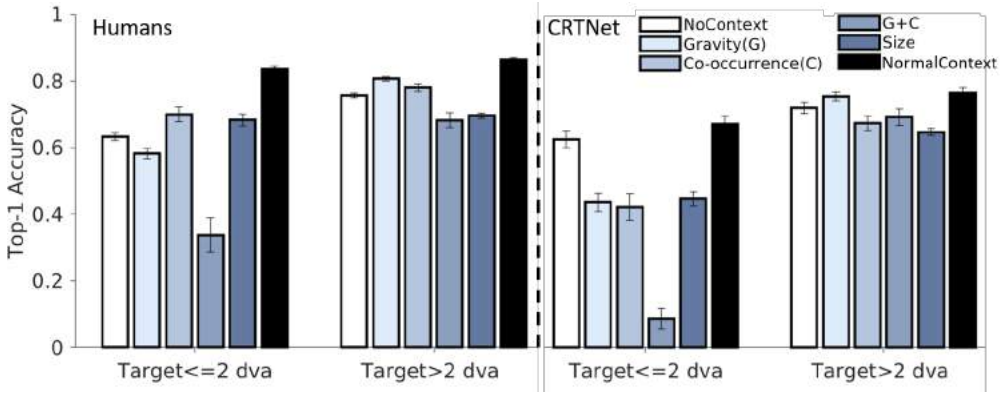



**Figure 6.3: Recognition accuracy of humans and CRTNet on our VirtualHome dataset.** All results shown are only for the 16 overlapping categories between VirtualHome and COCO-Stuff.

In order to assess CRTNet's performance, we decided to compare it against a set of baseline models - Faster R-CNN (Ren et al., 2015), DenseNet (Huang et al., 2017), and CATNet (Zhang et al., 2020) (see Sec. 2 for more information on the models). We start by testing the computational models on our VirtualHome data. To do that, we trained all of them on the 16 overlapping categories from COCO-Stuff (Caesar et al., 2018). We tested the models on all 12 VirtualHome conditions (6

contextual ones, split into 2, based on object size) as described in Section 4.1. Human and CRTNet performances on the 16 overlapping categories are summarized in Figure 6.3, while the performance of the baselines is summarized in Figure 6.4. We note the following facts about CRTNet's performance:

1. **Normal Context helps recognition.** Consistent with previous research and our findings in Sec. 6.1, in-context objects are significantly easier to recognize then out-of-context objects. This is especially true for smaller objects, since contextual information matters more for them (see Sec. 6.1).

2. **No context is better than misleading context.** CRTNet relies on contextual cues to recognize objects, and thus when it is presented with misleading context, it generally performs worse than if there is no context. That being said, CRTNet exhibits a lot of robustness when compared to other models (see Sec. 6.3) and that is particularly apparent when recognizing big objects.

3. **CRTNet exhibits human-like recognition patterns.** All of the patterns of human recognition that we discussed in Sec. 6.1 are also apparent in CRTNet. Interestingly, the relative accuracy between conditions is highly preserved between human and CRTNet recognition. The model performs the worst at the combination of gravity and co-occurrence violation condition, followed by the other out-of-context conditions showing little to none difference in accuracy. Both humans and CRTNet recognize big objects much more easily than smaller ones (even though the difference between those conditions is much larger in CRTNet). Remarkably, the linear correlation between human and CRTNet performance is the highest out of all baselines - 0.89 (Table 6.1).

4. **CRTNet performs generally worse than humans.** CRTNet shows lower accuracy on

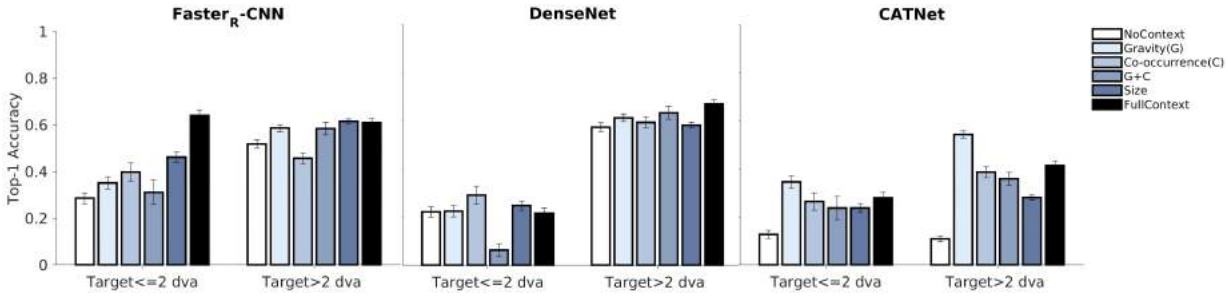



**Figure 6.4: Recognition accuracy of the baseline models - Faster R-CNN, DenseNet, and CATNet on our VirtualHome dataset.** All results shown are for the 16 overlapping categories between VirtualHome and COCO-Stuff.

| VirtualHome Dataset | Overall |
|---|---|
| CRTNet (ours) | **0.89** |
| Faster R-CNN (Ren et al., 2015) | 0.73 |
| DenseNet (Huang et al., 2017) | 0.66 |
| CATNet (Zhang et al., 2020) | 0.36 |

**Table 6.1: Linear correlations between human and model performance over the 12 contextual conditions.**

all conditions even though it performs almost on par with humans when recognizing big objects. Humans are still much better at recognizing small objects.

5. **CRTNet surpasses baseline accuracy on most conditions.** CRTNet achieves significantly higher accuracy then Faster R-CNN and CATNet when tested on big objects. The model also performs much better than DenseNet when tested on small objects. That being said, Faster R-CNN and CATNet still outperform CRTNet on the combination of gravity and co-occurrence condition.

## 6.3 Computational Model Recognition in Cut-and-paste Data

We also wanted to determine how the computational models perform on the cut-and-paste dataset (Zhang et al., 2020) and gain insight into how they deal with different types of contextual information. To do that, we again trained all of them on the 55 overlapping categories from the COCO-

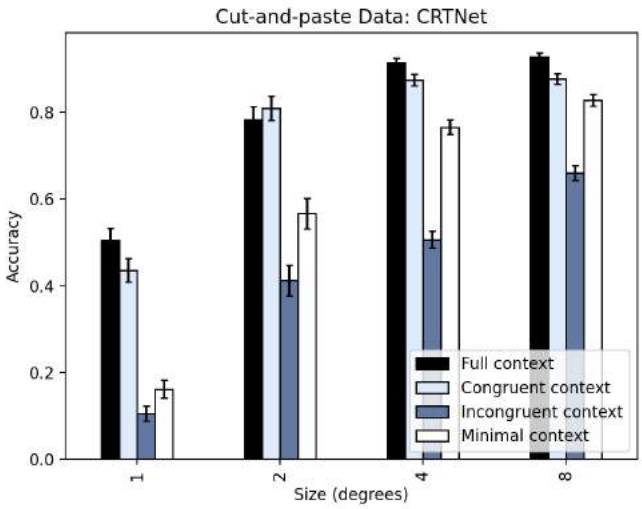



**Figure 6.5: CRTNet's performance on cut-and-paste data.** This figure shows top-1 accuracy on all 16 conditions from the cut-and-paste dataset. Full context improves CRTNet's accuracy while incongruent context impairs it. Congruent context helps model's recognition more than minimal context.

Stuff dataset (Caesar et al., 2018). We tested the models on all 16 conditions in the cut-and-paste dataset as described in Section 4.2.

CRTNet's accuracy on all of the conditions is summarized in Figure 6.5. Complete results of humans and baseline models can be seen in Table 6.2 (alternatively, see Fig. 6.6 for the same data organized as barplots). We can make the following observations about CRTNet's performance:

1. **Congruent context improves performance.** Unsurprisingly, full contextual information helps recognition for all models that integrate some contextual information when making a label prediction. The same effect can also be seen in the congruent condition which is a little bit less helpful than the original context as one might expect. We see similar results in humans and in all computational models.

2. **Less context is better than misleading context.** All models perform worse on incongruent context than on any other condition, including minimal context. This is even seen in

DenseNet which does not explicitly use contextual information even though when the target object is cropped, some background pixels are also being processed with it (due to cropping the bounding box of the target and not necessarily the target object itself). This suggests that even a few misleading pixels might hurt the model's performance.

3. **CRTNet often performs better than humans.** CRTNet achieves higher accuracy than humans in almost every condition. Moreover, in many conditions it beats the human performance by more than 10%. However, humans are still better at recognizing small objects. It is also worth noting that in these experiments, the human subjects had very limited viewing time which contributed to their lower results, while the models were not time capped.

4. **CRTNet has a superior performance compared to baselines.** CRTNet has a significantly higher accuracy than other models. The only condition in which other baseline model performs better is on incongruent context where DenseNet achieves higher accuracy for two of the size bins. This makes sense, since the best strategy when recognizing objects in incongruent context would be to completely ignore the context which is exactly what DenseNet does.

5. **CRTNet's attention module helps recognition even in the absence of contextual information.** Originally, we included the attention module in order to improve integration of contextual and target information. However, when comparing DenseNet and CRTNet on the minimal context condition, it is apparent that the attention module helps recognition even in the absence of context, since that is the only difference in the models (see Sec. 3.5 for discussion).

| | Size [0.5, 1] dva | | | | Size [1.75, 2.25] dva | | | | Size [3.5, 4.5] dva | | | | Size [7, 9] dva | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FC | CG | IG | MC | FC | CG | IG | MC | FC | CG | IG | MC | FC | CG | IG | MC |
| Humans | | | | | | | | | | | | | | | | |
| (Zhang et al., 2020) | **56.0** | 18.8 | 5.9 | 10.1 | 66.8 | 48.6 | 22.3 | 38.9 | 78.9 | 66.0 | 38.8 | 62.0 | 88.7 | 70.7 | 59.0 | 77.4 |
| | (2.8) | (2.3) | (1.3) | (1.7) | (2.7) | (2.8) | (2.4) | (2.8) | (2.4) | (2.7) | (2.6) | (2.8) | (1.7) | (2.6) | (2.8) | (2.3) |
| CRTNet | | | | | | | | | | | | | | | | |
| (ours) | 50.2 | **43.9** | 10.6 | **17.4** | **78.4** | **81.4** | **41.2** | **56.7** | **91.5** | **87.3** | 51.1 | **76.6** | **92.9** | **87.7** | **66.4** | **83.0** |
| | (2.8) | (2.8) | (1.7) | (2.1) | (3.0) | (2.8) | (3.5) | (3.6) | (1.1) | (1.3) | (1.9) | (1.6) | (0.9) | (1.2) | (1.7) | (1.4) |
| Faster R-CNN | | | | | | | | | | | | | | | | |
| (Ren et al., 2015) | 24.9 | 10.9 | 5.9 | 7.2 | 44.3 | 27.3 | 20.1 | 16.5 | 65.1 | 53.2 | 39.0 | 42.9 | 71.5 | 64.3 | 55.0 | 64.6 |
| | (2.4) | (1.7) | (1.3) | (1.4) | (3.6) | (3.2) | (2.9) | (2.7) | (1.8) | (1.9) | (1.9) | (1.9) | (1.6) | (1.7) | (1.8) | (1.7) |
| DenseNet | | | | | | | | | | | | | | | | |
| (Huang et al., 2017) | 13.1 | 10.0 | **11.2** | 12.5 | 45.4 | 42.3 | 39.7 | 46.4 | 67.1 | 62.3 | **55.4** | 67.1 | 74.9 | 67.2 | 63.5 | 74.9 |
| | (1.9) | (1.7) | (1.8) | (1.8) | (3.6) | (3.5) | (3.5) | (3.6) | (1.8) | (1.9) | (1.9) | (1.8) | (1.6) | (1.7) | (1.7) | (1.6) |
| CATNet | | | | | | | | | | | | | | | | |
| (Zhang et al., 2020) | 37.5 | 29.2 | 3.6 | 6.1 | 53.0 | 46.5 | 10.9 | 22.1 | 72.8 | 71.2 | 24.5 | 38.9 | 81.8 | 78.9 | 47.6 | 74.8 |
| | (4.0) | (2.4) | (1.0) | (2.0) | (4.1) | (2.5) | (1.6) | (3.6) | (3.6) | (2.4) | (2.2) | (3.9) | (3.0) | (2.1) | (2.6) | (3.5) |

**Table 6.2: Recognition accuracy of humans, our model (CRTNet, and baselines), Faster R-CNN, DenseNet, and CATNet on the cut-and-paste dataset (Zhang et al., 2020).** All images are split into 4 bins based on size and there are 4 conditions for each size: full context (FC), congruent context (GC), incongruent context (IC) and minimal context (MC) (see Sec. 4.2). Bold highlights the best performance. Numbers in brackets denote standard error of the mean.

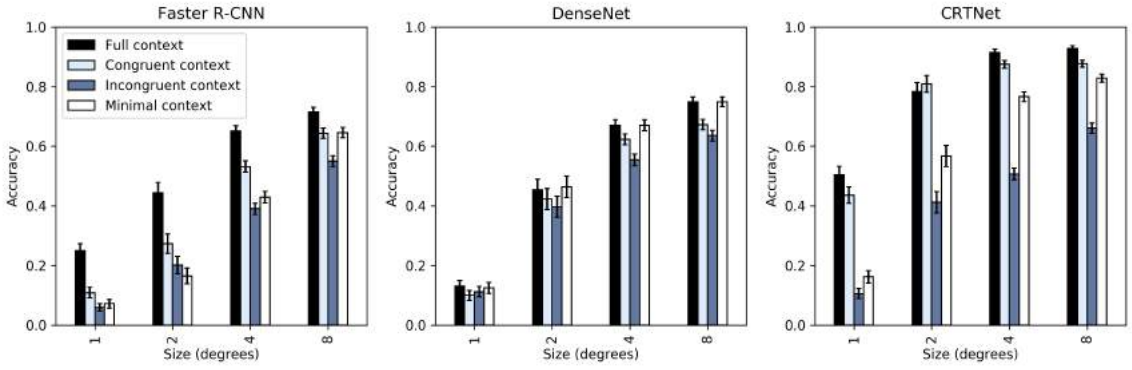



**Figure 6.6: Barplots of the cut-and-paste results for Faster R-CNN, DenseNet, and CRTNet.**

# 7

# Discussion

## 7.1 Comparison between CRTNet and Baseline Models

In this section, we aim to continue the discussion from Sec. 6.3 and examine how the novel model CRTNet compares to our state-of-the-art deep learning baseline models.

### 7.1.1 Faster R-CNN

Faster R-CNN has the simplest architecture out of our baseline models, consisting only of a feature extractor, ROI pooling layer, and multiple fully-connected layers (Ren et al., 2015). However, the model manages to integrate some global contextual information in its feature maps through the processing done in its CNN which we hypothesized to be useful in normal or congruent context.

Nevertheless, its performance falls short when compared to CRTNet. From the two experiments that we designed, Faster R-CNN managed to beat CRTNet only in one condition - the combination of gravity and object co-occurrences contextual condition for small objects in VirtualHome. We hypothesize that the worse results of Faster R-CNN can be attributed to its relatively weak feature extractor. Even though it manages to incorporate some contextual information, a feature extractor such as VGG16 (Simonyan and Zisserman, 2014) has been shown to be less effective than ResNet (He et al., 2016) and DenseNet (Huang et al., 2017).

### 7.1.2 DenseNet

DenseNet has shown very promising results on the task of object recognition (Huang et al., 2017). Its main strength lies in its powerful feature extractor that allows for a variety of wanted qualities such as bigger feature maps, recycled features, and easier training. However, DenseNet crops the target image before processing, removing (almost) all contextual information, which we hypothesized to be a weakness (Huang et al., 2017).

Due to these qualities of DenseNet, we can see that the model is unaffected by most of our conditions in both datasets. In the cut-and-paste dataset, DenseNet achieves the same accuracy for both minimal and full context, while in the VirtualHome dataset, it achieves even higher results in co-occurrence condition than in the normal one. While this allowed the model to have higher accuracy than CRTNet in two of the incongruent conditions, DenseNet's performance was worse in all other

conditions where a model could benefit from some contextual information (e.g. DenseNet scored more than 20% lower in the full context conditions; see Table 6.2).

### 7.1.3 CATNet

CATNet integrates contextual information in a human-like way by using attention mechanisms and a recurrent memory (Zhang et al., 2020). The model uses the same feature extractor as Faster R-CNN (i.e. VGG16 (Simonyan and Zisserman, 2014)), which we identified as a potential weakness.

Similarly to Faster R-CNN, the model's accuracy fell short when compared to CRTNet. Surprisingly, CATNet was also least correlated with human performance (Table 6.1).

## 7.2 Conclusion

We quantitatively and systematically studied the role of context in visual recognition in humans and computational models. We introduced a novel dataset based on the VirtualHome engine (Puig et al., 2018) that consists of 15,773 images of synthetic 3D indoor scenes. The VirtualHome dataset allowed us to systematically study 6 contextual conditions - normal context, gravity violation, object co-occurrences violation, size regularities violation, combination of gravity and object co-occurrences violation, and no context. We also used the cut-and-paste dataset (Zhang et al., 2020) of real photographs designed specifically for studying the effects of contextual information on object recognition.

We proposed an original supervised machine learning model - the Context-aware Recognition Transformer Network (CRTNet) that integrates contextual and object information in a novel robust way. CRTNet consists of two streams - a context stream and a target stream. The context stream processes and integrates contextual and target cues via multi-head transformer decoding layers. The target stream processes only the target object and uses a confidence estimator to combine

the information from both streams.

To assess our model's performance we implemented 3 baseline architectures - Faster R-CNN (Ren et al., 2015), DenseNet (Huang et al., 2017), and CATNet (Zhang et al., 2020). Additionally, as an essential benchmark, we also measured human's performance on the task in a series of psychophysics experiments. Our model showed superior performance over competitive baselines and even over humans in a wide range of conditions. Despite this great success, we also have to note that humans still achieve significantly higher accuracy when recognizing small objects.

Ultimately, we discussed various ways in which context influences object recognition and we examined what parts of a computational model seem to be most useful for robust object recognition.

However, there is still a lot more work in the field of out-of-context object recognition that needs to be done. Future studies might focus on learning more about other contextual conditions that we did not study in this thesis such as luminescence, color, and material of objects. Future computational models should focus on finding more accurate ways of recognizing small objects, whereas further neuroscience studies may use the human recognition artefacts that we listed in Section 6.1 in order find neural correlates for contextual reasoning in the brain.

# A

# Tables for Generating In-Context Objects

**Table A.1: Defining Natural Context for Objects: Surfaces.** On what surfaces would you expect to see the given objects (e.g. the bathroom counter is unnatural location for apples, but not for bar soaps). A natural location is indicated by "True" in the table; An atypical location is indicated by "False".

| object | bathroom counter | bench | book shelf | coffee table | desk | dish washer | kitchen counter | kitchen table | night stand | stove | tv stand | washing machine |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| apple | False | True | True | True | True | False | True | True | False | True | True | False |

**Table A.1: Defining Natural Context for Objects: Surfaces.** On what surfaces would you expect to see the given objects (e.g. the bathroom counter is unnatural location for apples, but not for bar soaps). A natural location is indicated by "True" in the table; An atypical location is indicated by "False".

| object | bathroom counter | bench | book shelf | coffee table | desk | dish washer | kitchen counter | kitchen table | night stand | stove | tv stand | washing machine |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| barsoap | True | False | False | False | False | False | False | True | False | True | False | True |
| book | False | True | True | True | True | True | True | True | True | True | True | False |
| candle | True | True | False | True | True | False | False | True | True | True | True | True |
| cellphone | True | False | False | True | False | False | False | True | True | True | True | True |
| cereal | False | True | False | True | True | False | True | True | True | True | True | False |
| chinesefood | False | True | False | True | True | False | False | True | True | True | True | False |
| chocolatesyrup | False | True | True | True | True | False | False | True | False | True | True | False |
| coffeemaker | False | True | False | False | False | False | True | True | False | False | False | False |
| condimentbottle | False | True | True | True | True | False | True | True | False | True | True | False |
| condimentshaker | False | True | True | True | False | False | True | True | True | True | True | False |
| cupcake | False | True | True | True | True | False | True | True | False | True | True | False |
| cutleryknife | False | False | False | True | True | False | False | True | False | True | True | False |
| cutlets | False | False | False | False | False | False | True | True | False | True | False | False |
| dishbowl | False | False | True | True | True | False | True | True | True | True | True | False |
| dishwashingliquid | True | True | False | False | False | False | False | True | False | True | False | True |
| keyboard | False | False | False | True | True | False | False | False | True | False | True | False |
| lime | False | False | True | True | True | False | True | True | True | True | True | False |
| microwave | False | True | False | True | True | False | True | True | False | False | True | False |
| milkshake | False | True | True | True | True | False | True | True | True | True | True | False |
| mouse | False | False | False | True | True | False | False | False | True | False | True | False |

**Table A.1: Defining Natural Context for Objects: Surfaces.** On what surfaces would you expect to see the given objects (e.g. the bathroom counter is unnatural location for apples, but not for bar soaps). A natural location is indicated by "True" in the table; An atypical location is indicated by "False".

| object | bathroom counter | bench | book shelf | coffee table | desk | dish washer | kitchen counter | kitchen table | night stand | stove | tv stand | washing machine |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| mug | True | True | True | True | True | False | True | True | True | False | True | True |
| peach | False | False | True | True | False | False | True | True | True | True | True | False |
| pie | False | False | True | True | True | False | True | True | False | True | True | False |
| pillow | False | False | False | True | False | False | False | False | False | False | False | False |
| plate | True | False | False | True | True | False | False | True | True | True | True | True |
| plum | False | False | True | True | True | False | True | True | True | True | True | False |
| poundcake | False | True | True | True | True | False | False | True | False | True | True | False |
| pudding | False | False | True | True | True | False | True | True | True | True | True | False |
| remotecontrol | False | False | False | True | True | False | False | False | True | False | True | False |
| slippers | True | False | False | True | False | False | False | True | True | False | True | True |
| toothbrush | True | False | False | False | False | False | False | False | False | False | False | True |
| toothpaste | True | False | False | False | False | False | False | False | False | False | False | True |
| towel | True | True | True | True | True | False | True | True | True | True | True | True |
| washingsponge | False | True | False | False | False | False | True | True | False | True | False | False |
| wineglass | False | True | True | True | True | False | False | True | True | False | True | False |

**Table A.2: Defining Natural Context for Objects: Rooms.** In what rooms would you expect to see the given objects (e.g. the bathroom is unnatural location for apples, but not for bar soaps). A natural location is indicated by "True" in the table; An atypical location is indicated by "False".

| object | bathroom | bedroom | kitchen | livingroom |
|---|---|---|---|---|
| apple | False | True | True | True |
| barsoap | True | False | True | False |
| book | False | True | True | True |
| candle | True | True | True | True |
| cellphone | True | True | True | True |
| cereal | False | False | True | True |
| chinesefood | False | True | True | True |
| chocolatesyrup | False | False | True | True |
| coffeemaker | False | False | True | False |
| condimentbottle | False | False | True | True |
| condimentshaker | False | False | True | True |
| cupcake | False | True | True | True |
| cutleryknife | False | False | True | True |
| cutlets | False | False | True | False |
| dishbowl | False | True | True | True |
| dishwashingliquid | True | False | True | False |
| keyboard | False | True | False | True |
| lime | False | True | True | True |
| microwave | False | False | True | True |
| milkshake | False | True | True | True |
| mouse | False | True | False | True |

**Table A.2: Defining Natural Context for Objects: Rooms.** In what rooms would you expect to see the given objects (e.g. the bathroom is unnatural location for apples, but not for bar soaps). A natural location is indicated by "True" in the table; An atypical location is indicated by "False".

| object | bathroom | bedroom | kitchen | livingroom |
|---|---|---|---|---|
| mug | True | True | True | True |
| peach | False | True | True | True |
| pie | False | False | True | True |
| pillow | False | True | True | True |
| plate | True | True | True | True |
| plum | False | True | True | True |
| poundcake | False | True | True | True |
| pudding | False | False | True | True |
| remotecontrol | False | True | False | True |
| slippers | True | True | True | True |
| toothbrush | True | False | False | False |
| toothpaste | True | False | False | False |
| towel | True | True | True | True |
| washingsponge | False | False | True | False |
| wineglass | False | True | True | True |

# B

# Graphical User Interface for Filtering Images

```
from PIL import Image
from easygui import *
from tkinter import Tk
import os.path
import glob
import sys

msg = "What do you want to do?"
```

```
title = "Image Selector GUI"
choices = ["Classify Images", "See Good Images", "See Bad Images"]
choice = choicebox(msg, title, choices)

if choice == "See Good Images":
    # Choose an apartment
    msg = "Which apartment do you want to see?"
    title = "Choose an apartment"
    choices = ["0", "1", "2", "3", "4", "5", "6", "other"]
    choice = choicebox(msg, title, choices)
    apartment_name = "apartment_" + str(choice)

    # Load all good images from that apartment
    file_good = open(apartment_name + "_good.txt", "r")
    good_images = file_good.readlines()

    if good_images == []:
        msg = msgbox("You need to have a file named " + apartment_name + "_good.txt in order to use this option.", "Warning")
        sys.exit()

    # Remove newlines
    good_images = map(lambda s: s.strip(), good_images)
    good_images_list  = list(good_images)

    # Show all of the images
    i = 0
    while i < len(good_images_list):
        filename = good_images_list[i]

        output = buttonbox("", "Good Images: " + filename, image = filename, choices = ["Back", "Next", "Copy", "Cancel"])
        if output == "Cancel":
            break
        elif output == "Back":
            i -= 1
            continue
        elif output == "Next":
            i += 1
            continue
        elif output == "Copy":
            r = Tk()
            r.withdraw()
            r.clipboard_clear()
            r.clipboard_append(filename)
            r.update() # now it stays on the clipboard after the window is closed
            r.destroy()

elif choice == "See Bad Images":
    # Choose an apartment
    msg ="Which apartment do you want to see?"
    title = "Choose an apartment"
    choices = ["0", "1", "2", "3", "4", "5", "6", "other"]
    choice = choicebox(msg, title, choices)
```

```
    apartment_name = "apartment_" + str(choice)

    # Load all bad images from that apartment
    file_bad = open(apartment_name + "_bad.txt", "r")
    bad_images = file_bad.readlines()

    if bad_images == []:
        msg = msgbox("You need to have a file named " + apartment_name + "_bad.txt in order to use this option.", "Warning")
        sys.exit()

    # Remove newlines
    bad_images = map(lambda s: s.strip(), bad_images)
    bad_images_list  = list(bad_images)

    # Show all of the images
    i = 0
    while i < len(bad_images_list):
        filename = bad_images_list[i]

        output = buttonbox("", "Bad Images: " + filename, image = filename, choices = ["Back", "Next", "Copy", "Cancel"])
        if output == "Cancel":
            break
        elif output == "Back":
            i -= 1
            continue
        elif output == "Next":
            i += 1
            continue
        elif output == "Copy":
            r = Tk()
            r.withdraw()
            r.clipboard_clear()
            r.clipboard_append(filename)
            r.update() # now it stays on the clipboard after the window is closed
            r.destroy()

elif choice == "Classify Images":
    # Choose an apartment
    msg ="Which apartment do you want to review?"
    title = "Choose an apartment"
    choices = ["0", "1", "2", "3", "4", "5", "6", "other"]
    choice = choicebox(msg, title, choices)
    apartment_name = "apartment_" + str(choice)

    # Load all of the images from that apartment
    all_images = glob.glob(apartment_name + '/*.png')

    if all_images == []:
        msg = msgbox("You need to have a folder named " + apartment_name + " in order to use this option.", "Warning")
        sys.exit()
```

```
    # Prepare the buttonbox
    text = "Should this image be included in the dataset"
    title = "Image Selector"
    button_list = ["Good", "Bad", "Cancel"]
    total, good_cnt, bad_cnt = 0, 0, 0

    # create apartment_name_good.txt and apartment_name_bad.txt if they don't exist
    file_exists = os.path.isfile(apartment_name + "_good.txt")
    if not file_exists:
        file_good = open(apartment_name + "_good.txt", "w")
        file_good.close()

    file_exists = os.path.isfile(apartment_name + "_bad.txt")
    if not file_exists:
        file_good = open(apartment_name + "_bad.txt", "w")
        file_good.close()

    # Make a set of all of the images that have already been processes
    file_good = open(apartment_name + "_good.txt", "r")
    file_bad = open(apartment_name + "_bad.txt", "r")

    already_processed = set()
    for filename in file_good:
        already_processed.add(filename.strip())
    for filename in file_bad:
        already_processed.add(filename.strip())

    file_good.close()
    file_bad.close()

    num_img = 0

    # Go through all of the images in the chosen apartment
    for filename in all_images:

        num_img += 1
        print("Processing image: " + str(num_img) + "/" + str(len(all_images)))

        # Don't process if it has been already filtered
        if filename in already_processed:
            continue

        button_not_clicked = True
        cancel = False

        # Open files to write the results in
        file_good = open(apartment_name + "_good.txt", "a")
        file_bad = open(apartment_name + "_bad.txt", "a")

        while(button_not_clicked):
            # Resize in order to be able to always fit on the screen
            im = Image.open(filename)
            newsize = (600, 400)
            im = im.resize(newsize)
```

```
            im = im.save("resized.png")

            # Open button box and wait for user response
            output = buttonbox("", "Classify Images: " + filename, image = "resized.png", choices =
    button_list)
            button_not_clicked = False
            if output == "Good":
                good_cnt += 1
                file_good.write(filename + '\n')

            elif output == "Bad":
                bad_cnt += 1
                file_bad.write(filename + '\n')

            elif output == "Cancel":
                cancel = True

            else:
                button_not_clicked = True
                msg = msgbox("Please select one of the buttons!", "Try again")

        # Save whatever is written
        file_good.close()
        file_bad.close()

        if cancel:
            break
        else:
            total += 1

    # Show summary statistics after going through all of the images
    title = "Summary Statistics"
    message = "Total number of images processed: " + str(total) + "\n" + "Number of good images: " + str(
    good_cnt) + "\n" + "Number of bad images: " + str(bad_cnt) + "\n" + "Ratio accepted: " + str(good_cnt/
    total)
    msg = msgbox(message, title)
```

# References


Adelson EH, Bergen JR (1985) Spatiotemporal energy models for the perception of motion. *Josa a* 2:284–299.

Agrawal P, Nair A, Abbeel P, Malik J, Levine S (2016) Learning to poke by poking: Experiential learning of intuitive physics. *arXiv preprint arXiv:1606.07419* .

Albawi S, Mohammed TA, Al-Zawi S (2017) Understanding of a convolutional neural network In *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6. Ieee.

Alom MZ, Taha TM, Yakopcic C, Westberg S, Sidike P, Nasrin MS, Van Esesn BC, Awwal AAS, Asari VK (2018) The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164* .

Ba J, Mnih V, Kavukcuoglu K (2014) Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755* .

Bar M (2004) Visual objects in context. *Nature Reviews Neuroscience* 5:617–629.

Bar M, Ullman S (1996) Spatial context in recognition. *Perception* 25:343–352.

Baron-Cohen S, Wheelwright S, Spong A, Scahill V, Lawson J et al. (2001) Are intuitive physics and intuitive psychology independent? a test with children with asperger syndrome. *Journal of developmental and learning disorders* 5:47–78.

Battaglia P, Ullman T, Tenenbaum J, Sanborn A, Forbus K, Gerstenberg T, Lagnado D (2012) Computational models of intuitive physics In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 34.

Beery S, Van Horn G, Perona P (2018) Recognition in terra incognita In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 456–473.

Boussaoud D, Desimone R, Ungerleider LG (1991) Visual topography of area teo in the macaque. *Journal of comparative neurology* 306:554–575.

Brendel W, Bethge M (2019) Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760* .

Brincat SL, Connor CE (2004) Underlying principles of visual shape selectivity in posterior inferotemporal cortex. *Nature neuroscience* 7:880–886.

Bullier J (2001) Integrated model of visual processing. *Brain research reviews* 36:96–107.

Caesar H, Uijlings J, Ferrari V (2018) Coco-stuff: Thing and stuff classes in context In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1209–1218.

Choi MJ, Torralba A, Willsky AS (2012) Context models and out-of-context objects. *Pattern Recognition Letters* 33:853–862.

Conway BR (2018) The organization and operation of inferior temporal cortex. *Annual review of vision science* 4:381–402.

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee.

DiCarlo JJ, Zoccolan D, Rust NC (2012) How does the brain solve visual object recognition? *Neuron* 73:415–434.

Divvala SK, Hoiem D, Hays JH, Efros AA, Hebert M (2009) An empirical study of context in object detection In *2009 IEEE Conference on computer vision and Pattern Recognition*, pp. 1271–1278. IEEE.

Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S et al. (2020) An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* .

Dvornik N, Mairal J, Schmid C (2018) Modeling visual context is key to augmenting object detection datasets In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 364–380.

Felleman DJ, Van Essen DC (1991) Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)* 1:1–47.

Geirhos R, Janssen DH, Schütt HH, Rauber J, Bethge M, Wichmann FA (2017) Comparing deep neural networks against humans: object recognition when the signal gets weaker. *arXiv preprint arXiv:1706.06969* .

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

Girshick R (2015) Fast r-cnn In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.

Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.

Goodale MA, Milner AD (1992) Separate visual pathways for perception and action. *Trends in neurosciences* 15:20–25.

Grill-Spector K, Malach R (2004) The human visual cortex. *Annu. Rev. Neurosci.* 27:649–677.

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hochreiter S (1998) The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6:107–116.

Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural computation* 9:1735–1780.

Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.

Hung CP, Kreiman G, Poggio T, DiCarlo JJ (2005) Fast readout of object identity from macaque inferior temporal cortex. *Science* 310:863–866.

Jeannerod M, Arbib MA, Rizzolatti G, Sakata H (1995) Grasping objects: the cortical mechanisms of visuomotor transformation. *Trends in neurosciences* 18:314–320.

Kiani R, Esteky H, Mirpour K, Tanaka K (2007) Object category structure in response patterns of neuronal population in monkey inferior temporal cortex. *Journal of neurophysiology* 97:4296–4309.

Kravitz DJ, Saleem KS, Baker CI, Ungerleider LG, Mishkin M (2013) The ventral visual pathway: an expanded neural framework for the processing of object quality. *Trends in cognitive sciences* 17:26–49.

Kreiman G, Hung CP, Kraskov A, Quiroga RQ, Poggio T, DiCarlo JJ (2006) Object selectivity of local field potentials and spikes in the macaque inferior temporal cortex. *Neuron* 49:433–445.

Kreiman G, Serre T (2020) Beyond the feedforward sweep: feedback computations in the visual cortex. *Annals of the New York Academy of Sciences* 1464:222.

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25:1097–1105.

Kruger N, Janssen P, Kalkan S, Lappe M, Leonardis A, Piater J, Rodriguez-Sanchez AJ, Wiskott L (2012) Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE transactions on pattern analysis and machine intelligence* 35:1847–1871.

Kubricht JR, Holyoak KJ, Lu H (2017) Intuitive physics: Current research and controversies. *Trends in cognitive sciences* 21:749–759.

LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. *Neural computation* 1:541–551.

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86:2278–2324.

Liao YH, Puig X, Boben M, Torralba A, Fidler S (2019) Synthesizing environment-aware activities via activity sketches In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: Common objects in context In *European conference on computer vision*, pp. 740–755. Springer.

Logothetis NK, Sheinberg DL (1996) Visual object recognition. *Annual review of neuroscience* 19:577–621.

McCloskey M (1983) Intuitive physics. *Scientific american* 248:122–131.

McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5:115–133.

Merigan WH, Eskin TA (1986) Spatio-temporal vision of macaques with severe loss of p$\beta$ retinal ganglion cells. *Vision Research* 26:1751–1761.

Nakamura H, Gattass R, Desimone R, Ungerleider LG (1993) The modular organization of projections from areas v1 and v2 to areas v4 and teo in macaques. *Journal of Neuroscience* 13:3681–3691.

Orban GA (2008) Higher order visual processing in macaque extrastriate cortex. *Physiological reviews* .

Potter MC (1976) Short-term conceptual memory for pictures. *Journal of experimental psychology: human learning and memory* 2:509.

Puig X, Ra K, Boben M, Li J, Wang T, Fidler S, Torralba A (2018) Virtualhome: Simulating household activities via programs In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502.

Puig X, Shu T, Li S, Wang Z, Tenenbaum JB, Fidler S, Torralba A (2020) Watch-and-help: A challenge for social perception and human-ai collaboration.

Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497* .

Rentzeperis I, Kiper DC (2010) Evidence for color and luminance invariance of global form mechanisms. *Journal of Vision* 10:6–6.

Roelfsema PR, Lamme VA, Spekreijse H (2000) The implementation of visual routines. *Vision research* 40:1385–1411.

Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* 65:386.

Rosenfeld A, Zemel R, Tsotsos JK (2018) The elephant in the room. *arXiv preprint arXiv:1808.03305* .

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. *nature* 323:533–536.

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M et al. (2015) Imagenet large scale visual recognition challenge. *International journal of computer vision* 115:211–252.

Rust NC, DiCarlo JJ (2010) Selectivity and tolerance ("invariance") both increase as visual information propagates from cortical area v4 to it. *Journal of Neuroscience* 30:12978–12995.

Saleem K (1992) Pha-l study of connections from teo and v4 to te in the monkey visual cortex In *Society for Neuroscience Abstracts*, Vol. 18, p. 294.

Saleem K, Suzuki W, Tanaka K, Hashikawa T (2000) Connections between anterior inferotemporal cortex and superior temporal sulcus regions in the macaque monkey. *Journal of Neuroscience* 20:5083–5101.

Santurkar S, Tsipras D, Ilyas A, Madry A (2018) How does batch normalization help optimization? *arXiv preprint arXiv:1805.11604* .

Sáry G, Vogels R, Orban GA (1993) Cue-invariant shape selectivity of macaque inferior temporal neurons. *Science* 260:995–997.

Serban A, Poll E, Visser J (2020) Adversarial examples on object recognition: A comprehensive survey. *ACM Computing Surveys (CSUR)* 53:1–38.

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Singh KK, Mahajan D, Grauman K, Lee YJ, Feiszli M, Ghadiyaram D (2020) Don't judge an object by its context: Learning to overcome contextual bias In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11070–11078.

Sun J, Jacobs DW (2017) Seeing what is not there: Learning context to determine where objects are missing In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5716–5724.

Tanaka K (1996) Inferotemporal cortex and object vision. *Annual review of neuroscience* 19:109–139.

Thorpe S, Fize D, Marlot C (1996) Speed of processing in the human visual system. *nature* 381:520–522.

Tompa T, Sáry G (2010) A review on the inferior temporal cortex of the macaque. *Brain research reviews* 62:165–182.

Tong K, Wu Y, Zhou F (2020) Recent advances in small object detection based on deep learning: A review. *Image and Vision Computing* 97:103910.

Torralba A, Murphy KP, Freeman WT, Rubin MA (2003) Context-based vision system for place and object recognition In *Computer Vision, IEEE International Conference on*, Vol. 2, pp. 273–273. IEEE Computer Society.

Turk AM (2012) Amazon mechanical turk. *Retrieved August* 17:2012.

van Polanen V, Davare M (2015) Interactions between dorsal and ventral streams for controlling skilled grasp. *Neuropsychologia* 79:186–191.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. *arXiv preprint arXiv:1706.03762* .

Vogels R, Biederman I (2002) Effects of illumination intensity and direction on object coding in macaque inferior temporal cortex. *Cerebral Cortex* 12:756–766.

von Bonin G (1947) The neocortex of macaca mulatta. *Monographs in Medical Sciences.* 5:136.

Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y (2015) Show, attend and tell: Neural image caption generation with visual attention In *International conference on machine learning*, p. 2048–2057.

Yamane Y, Carlson ET, Bowman KC, Wang Z, Connor CE (2008) A neural code for three-dimensional object shape in macaque inferotemporal cortex. *Nature neuroscience* 11:1352–1360.

Zaremba W, Sutskever I, Vinyals O (2014) Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks In *European conference on computer vision*, pp. 818–833. Springer.

Zhang M, Tseng C, Kreiman G (2020) Putting visual object recognition in context In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12985–12994.

Zoccolan D, Cox DD, DiCarlo JJ (2005) Multiple object response normalization in monkey inferotemporal cortex. *Journal of Neuroscience* 25:8150–8164.