A Convolutional Neural Network Model for Predicting a Product's Function, Given Its Form

Matthew L. Dering*
Computer Science And Engineering
Penn State University
University, PA 16802
Email: mld284@psu.edu

Conrad S. Tucker
Engineering Design and
Industrial Engineering
Penn State University
University, PA 16802
Email: ctucker4@psu.edu

ABSTRACT

Quantifying the ability of a digital design concept to perform a function currently requires the use of costly and intensive solutions such as Computational Fluid Dynamics. To mitigate these challenges, the authors of this work propose a deep learning approach based on 3-Dimensional Convolutions that predicts Functional Quantities of digital design concepts. This work defines the term Functional Quantity to mean a quantitative measure of an artifact's ability to perform a function. Several research questions are derived from this work: i) Are learned 3D Convolutions able to accurately calculate these quantities, as measured by rank, magnitude and accuracy? ii) What do the latent features (that is, internal values in the model) discovered by this network mean? iii) Does this work perform better than other deep learning approaches at calculating Functional Quantities? In the case study, a proposed network design is tested for its ability to predict several functions (Sitting, Storing Liquid, Emitting Sound, Displaying Images, and Providing Conveyance) based on test form classes distinct from training class. This study evaluates several approaches to this problem based on a common architecture, with the best approach achieving F Scores of > 0.9 *in 3 of the 5 functions identified. Testing trained models on novel input also yields accuracy as high as 98% for estimating rank of these functional quantities. This method is also employed to differentiate between decorative and functional head-wear, which yields an 84.4% accuracy and 0.786 precision.*

1 Introduction

This work considers the relationship between object form and object function. As technology advances, so has the prevalence and utility of 3-Dimensional digital artifacts. Printing these artifacts at home or in local shops has grown additive manufacturing to a $5.1 billion industry[1]. Printed artifacts can be designed by the user, or, commonly, found in a shared repository or community such as Thingiverse or GrabCAD, where users can download models. GrabCAD alone boasts over 1.5 Million 3D models [2] available for download. The availability of 3D artifact capturing software and hardware (such as

*Address all correspondence to this author.

[1] http://www.forbes.com/sites/tjmccue/2016/04/25/wohlers%2Dreport%2D2016%2D3d%2Dprinter%2Dindustry%2Dsurpassed%2D5%2D1%2Dbillion/

[2] https://grabcad.com/library

3-D scanners, or RGB-D Sensors such as the Microsoft Kinect) has further enabled the capture, sharing, and production of such artifacts around the world [1, 2].

However, just because a design "looks good" does not mean that it will achieve its intended functions or behaviors. While simulation models such as CFD use advanced numerical methods and algorithms to evaluate the feasibility of designs, they require extensive modeling expertise, advanced computing resources and are time consuming [3, 4]. While some of these publicly-available digital design models are decorative in nature, many of these artifacts are meant to serve a specific purpose or *function* that may be implied or explicitly stated by the designer. Formally, a *function* is defined as a purpose intended for an object. This work proposes a method that uses as input, a model file representing a possible artifact. A neural network analyzes this artifact, and as output, predicts how well it will perform each intended function. For example, some objects can be used for sitting. This network will predict how much force it will be able to withstand, when sat upon. These trained predictive models will provide insight from their latent variables as well. A latent variable is a variable contained within the model whose value may describe the forms and patterns that have the most predictive power.

Typically, design defines a product by its form, function, and behavior [5]. However, this work is limited to a study of form and function since behavior can be influenced by independent forces such as environment and time. For example, a cellular telephone's function of making phone calls can be influenced by environment (such as the availability of cellular service) as well as time (the battery loses power over time). This work defines the term *Functional Quantity* to mean a real value that indicates the ability of an artifact to function in a certain way. For example, the functional quantity of a car would be 60 MPH, the average speed of a car, and the functional quantity of a bottle would be 20 fluid ounces, the average size of a bottle. In contrast, the functional quantity of an airplane would be 500 MPH, and a flower pot would be 128 fluid ounces. Note that the units of functional quantity will differ by function and the magnitude will differ by class. These quantities are derived based on best available data of the average for each class and quantity type. This work models this relationship as a regression problem, that will estimate the relationship between an object's form, and an object's functional quantities. This is based on the assumption that many object functions are implied by object appearance. Using a neural network to estimate these values will answer the question "How well will form X perform function Y?" in real values, without the need to set up a complex simulation environment (for example, how much water can this tanker hold?). In order to differentiate this work from an object recognition task, the network is tested using entirely untrained classes (rather than a test set similar to the trained classes), demonstrating the ability of this model to learn commonalities within functions. Even as automated simulation tools become more user friendly, this approach could see applications in other settings as well. For example, this method could envision secondary use cases for a given design, or aide a mobile AI system in understanding nearby objects.

This work is organized as follows. This section provides an introduction and motivation for this work. Section 2 presents other work making use of deep learning for 3-Dimensional artifacts as well as research in the engineering design community that has investigated form and function mapping techniques. Section 3 describes how deep learning is applied to this mapping problem. A case study of several functions is presented in Section 4, the results of which can be seen in Section 5. Finally, the work concludes with discussion of the results and possible future work derived from this.

2 Related Work

2.1 Design Evaluation Methods

Since design is such a broad field, design analysis and quantification can take many different approaches. Chandrasegaran et al. present an overview of some of the challenges [6]. Among them are collaborative design, digital knowledge discovery and how these problems aid in function to form mapping. Chen et al. proposed a method of representing functionality and knowledge in a structured format, so that inter-disciplinary knowledge could be transferred more easily [7]. Building on this, Qi et al. introduced a method wherein cross disciplinary concepts could be easily synthesized together to solve problems [8]. Bhatt et al. applied a similarly formalized approach to the architectural discipline [9]. Townsend et al. investigated how form and function influence a consumer's opinion of a brand [10]. Kang and Tucker proposed an automated method for generating new product forms and concepts from unrelated domains [11]. Tseng et al. analyzed product forms, and evaluated how their appearance corresponded to consumer's impressions of them [12]. Roy et al. formalized a framework of mapping product specification to representations and behavior of these products [13]. Tsai et al. presented a rapid design system that employed a parametric approach to predict adjectives used to describe their test designs [14]. Sylcott and Cagan examined the aggregate preference of users for form vs function [15, 16]; however, their work did not consider using form as a predictor of function, and rather explored the trade-offs involved. Osborn et al. investigated parameterizing aesthetic preference by consumers [17]. Tseng and Cagan proposed using a neural network to model consumer judgement of form and coupled this with a model that helped accomplish some objectives [18]. Tseng and Cagan considered aspects of an object within a just one class of objects, cars. So while Tseng and Cagan do attempt to fuse form and function, no automated prediction of what specific functions are capable by a given form was performed. Rather, the emphasis was on stylistic analysis. Instead of automated concept generation, this work explores the inverse of automated concept generation. I.e., an automated method of mapping form to function. Cheong et al. also drew on biologically inspired design to determine which keywords and analogies were best suited for design inspiration [19]. While this work does consider a mapping between form and function, it does not make use of textual description as input, which is not always available or accurate in its description of the form. For example, a caption provided for a chair may include "beautifully designed chair with a leather backing", rather than "this chair is capable of withstanding X amount of force". Rather, the work presented here uses 3D models as input and as output, a quantitative assessment of an object's functional capabilities.

Each of these works presented above explore the relationship between form and function, primarily through aiding or analyzing the design process pre- or post-function-to-form mapping. Indeed, all of these works consider the function to form process to be a necessarily manual one, that must be performed by a human designer, and focus on how to formalize the process, either by enabling communication or organizing knowledge. In other words, there exists a knowledge gap in the area of automatically and qualitatively predicting a product's function based on its form. This work helps to fill this knowledge gap by investigating the predictive power of form to function, including which functions are best predicted, and what types of features best capture this relationship.

2.2 Descriptor Machine Based Learning

Machine learning has presented many different solutions to the issue of object recognition, and other prediction problems. Traditionally, prediction consists of two main steps: i) descriptor extraction and ii) descriptor classification. A descriptor is typically a vector representation of an input of a fixed size. Many algorithms for extracting 2-Dimensional image descriptors have been proposed, such as SIFT [20], SURF [21], ORB [22], and HOGs [23]. These algorithms translate an image into a *descriptor vector*, which is a vector of numbers representing the data of the image. For recognition tasks, these descriptor vectors should be similar for similar categories, and different from others. Similarly, there have been 3-dimensional descriptor extraction methods proposed, such as ESF or SHOT [24]. Each of these methods uses as input a combination shape and color information to generate a descriptor vector. Song et al. introduced a framework which combined unsupervised learning, metric learning, and user intervention which used minimal input from users to validate proposed model groupings [25]. These descriptor methods rely on a domain expert to recognize and quantify the aspects of input which best suit the specified task (e.g. image gradients for object recognition).

The second portion in the prediction process is the classifier. Some of the most popular classifiers include Naive Bayes, Nearest Neighbor, Support Vector Machines (SVM) [26], and Random Forest Classifiers [27]. Descriptor-classifier based methods share a similar issue, which is that a descriptor representation must be engineered by a domain expert and selected. Each of these descriptors possess unique qualities. For example, in the 3-Dimensional arena, some descriptors are used for orientation recognition, instead of object recognition. To overcome these limitations, this work proposes the use of a machine learning technique called Convolutional Neural Networks (CNNs) that eliminates the need for choosing and employing one of these hand engineered descriptors. These networks will first learn descriptor representations of input, and then determine the optimal method of classification, in an end-to-end process, so that feature and classifier are jointly learned.

2.3 Convolutional Neural Networks

CNNs have proven to be one of the most effective tools available for many 2D image related tasks. CNN based solutions consistently outperform others in the Imagenet object recognition challenge [28]. Due to neural networks flexibility, high performers in this challenge ([29, 30, 31]) often see their designs integrated into specialized applications. Similarly, large annotated 3D object datasets (ModelNet [32] and ShapeNet [33]) have enabled research into classifying 3D objects. Works such as those by Maturana and Scherer [34], Su et al. [35], and Shi et al. [36] have proposed different network designs that are used to classify objects within these datasets. While classification is a common application of these networks, related to the problem addressed by this work, none of these networks can directly infer an objects function, nor would they be effective when presented with new, unseen classes. Other works have used 3D CNNs to infer other attributes of objects such as viewpoint [37], object deformability [38], segmentation [39], and context based on scenes [40]. However, explicit prediction of quantified object function is not directly addressed by any existing work.

While these approaches employ a CNN with 3-Dimensional input to achieve some goal, typically the desired output is either a categorical classification (e.g., “this object is a chair VS a boat”) or as a more general 3-Dimensional descriptor (e.g., “a description of an object chair is similar to the descriptor of another chair and different from the descriptor of a boat”).

```
Data: CAD File CAD, with vertices V, faces F
Result: Voxel Grid G
G = 0 for all x, y, z;
for Location x, y, z in G do
    if x, y, z in V or F then
        G [x,y,z] = 1;
    end
end
```

Algorithm 1: CAD Voxelization Algorithm

The method proposed in this paper uses a network to determine the relationship between form and function, and to predict an objects function, given its form.

3 Method

The method presented in this work describes the process of evaluating an artifact's functional quantities, based on its form, starting with data collection, followed by model training, and testing, then evaluation.

3.1 Voxel Data Description

The input data used by this method are twofold, i) voxel object forms and ii) artifact functional quantities. Object forms are often found in various CAD files, that contain facets, edges, and vertices. However, this work instead considers a voxelized form of these artifacts due to the challenge of analyzing the relative spatial information of facets, edges and vertices in their raw format (i.e., which typically is in an ṠTL, ȮBJ or ȮFF file format). Thus, each artifact is expressed as a 3D voxel grid of size $n \times n \times n$, where a binary variable at a point (x, y, z) indicates whether that voxel is occupied in the object form. Since most CAD files are not found in this format, including those available in online repositories, a voxelization program is employed to convert to this format, using a simple grid-occupancy test. Convolutional Neural Networks expect dense and uniformly sized input, therefore it is necessary to convert these artifacts from a sparse format (mesh) to a dense one (voxel grid). An explanation of this method can be seen in Algorithm 1. Additionally, the data is augmented by rotating 12 times around the gravity axis.

Finally, each form (3D artifact) carries with it a label $c \in \mathbb{R}^p$ that consists of $p \geq 1$ real valued variables, each indicating how well an object can be expected to perform the p^{th} functionality. These functional quantities are derived from the best available data from these types of objects, averaged across these objects. For example, the average bottle contains 20 fluid ounces, and thus is used as the target output value for an input of a voxelized bottle.

3.2 Convolutional Neural Network Design

This section provides an overview of Neural Network components. Each network design will contain different neurons of varying design and activation function, as well as data processing layers such as dropouts. Each component will be summarized in the following subsections, followed by an overview of the design presented in this work.

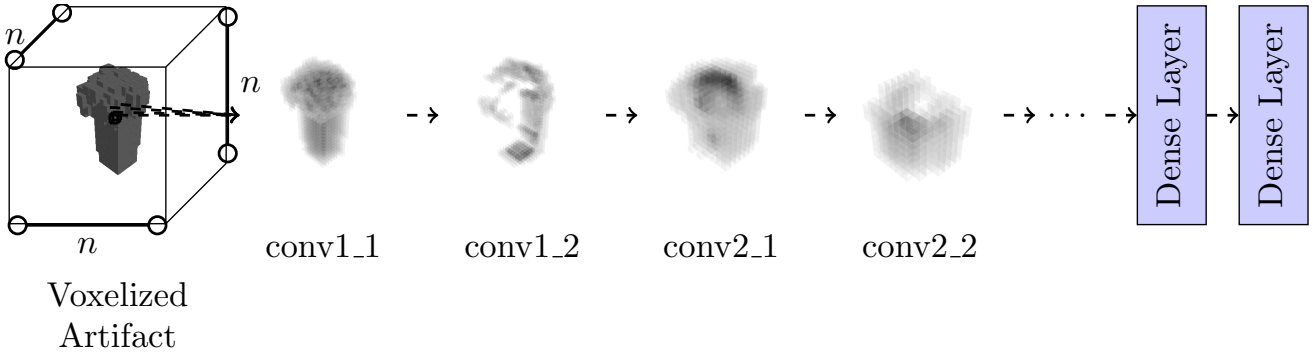


Fig. 1: Cross Sections of Selected Layers of a 3D CNN. These show the activations of certain kernels of the layers in the trained network on the voxelized input shown left

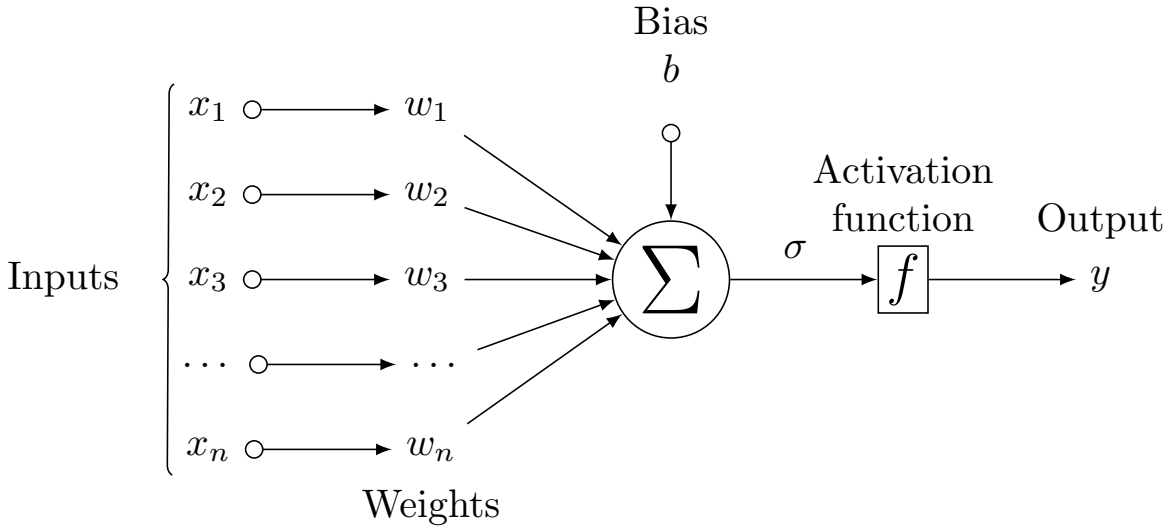


Fig. 2: An Artificial Neuron. The inputs are represented by x_i, that are multiplied by the weights w_i, summed with a bias term b and activated by a function f to produce an output y. Each layer type principally defines how the inputs are mapped to the previous layer, along with which activation function is employed. The rest of the terms are learned.

3.2.1 Artificial Neurons

The most basic unit of a neural network is the Artificial Neuron. An example of this neuron can be seen in Figure 2. A neuron accepts some number of inputs, multiplies each of them by a separate learned weight, and sums these products, along with a learned bias term. This sum then has an activation function applied to it such that $y = f(\sigma)$, where σ is the sum of the weighted inputs and bias, and y is the output of the neuron. These inputs $x_1 \cdots x_n$ are mapped to the input data of the network (at the bottom layer), or the output of the previous layer, for all other layers. Each layer is composed of many of these neurons. This is an iterative process; so, for example, the first layer of the neurons' inputs are connected to the raw input, (a binary variable indicating if the voxel is occupied), while the second layer of neurons is connected to the output of the previous layers. Lastly, each layer is made up of many of these neurons, and each layer is described primarily by how the inputs to each neuron are determined.

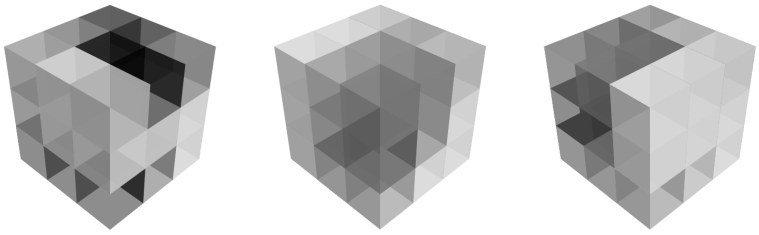


Fig. 3: Selected $3 \times 3 \times 3$ learned kernels

3.2.2 Activation Functions

Each layer has an activation function (f in Figure 2). This is a function that operates on the linear combination of inputs and bias, $\sigma = b + \sum_{i=1}^{n} x_i \times w_i$. This activation function can be any function with one input. For example, a common function is the linear function where $f(\sigma) = \sigma$. This work employs just two activation functions, i) linear activation and ii) Leaky Rectified Linear Unit or *LeakyReLU*. Linear activation returns σ for all values of σ.

LeakyReLUs are used in this work since they have been found to perform better in CNNs [41]. This function quantifies if and by how much a neuron is activated. This variation on the common *ReLU* non-linear function allows a small negative gradient to output if the neuron is not activated. The function is given by Equation 1:

$$f(\sigma) = \begin{cases} \sigma & \sigma > 0 \\ \alpha \cdot \sigma & \sigma \leq 0 \end{cases} \tag{1}$$

Where:

- σ is the sum, prior to activation from Figure 2 and
- α is a set parameter of the network. Note that for non-"leaky" ReLUs $\alpha = 0$

3.2.3 Dropout Layer

Dropout layers are a common Neural Network concept that help avoid overfitting by randomly setting some neuron activations to 0. These layers have a parameter that governs what percentage of activations are set to 0. These layers help avoid overfitting by ensuring that the same input will result in slightly different output for intermediate levels. The parameters for these regularizers are set at runtime, and are chosen to add up to approximate 0.5, so that half the neurons are reset in aggregate over the course of the network.

3.2.4 3D Convolutional Layers

The major component of these CNNs are layers called Convolutional Layers. Each of these layers C contains a set of n kernels (or filters), whose weights are learned. Each Convolutional kernel has a dimension, $d \times d \times d$ in size. Finally, each kernel in the layer contains a set of $d \times d \times d$ weights. An example of these weights can be seen in Figure 3. Briefly, these

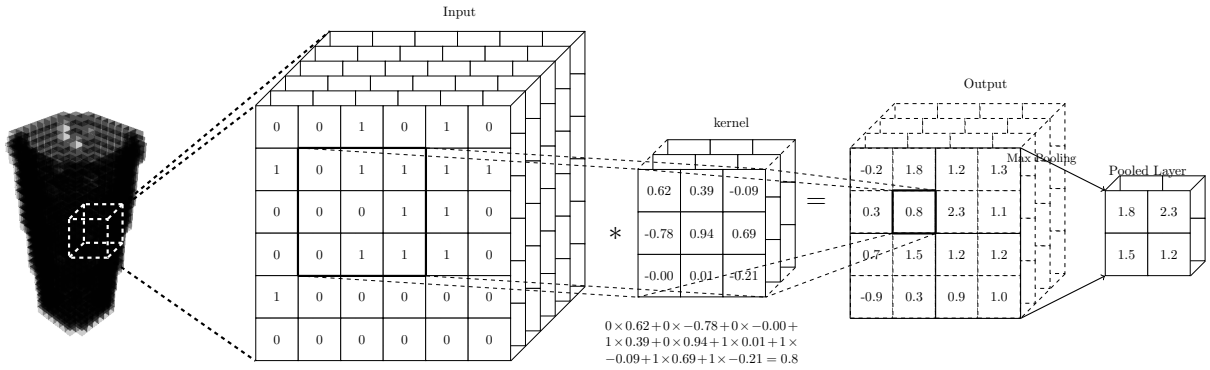


Fig. 4: The convolution and pooling operation

weights correspond to small regions, 3 voxels along each edge in size. The lower values, indicated by a darker cube, respond well when the voxel in this position is unoccupied. Similarly, the higher values, indicated by a lighter cube, will activate well when these regions are occupied. For example, the top kernel shown in Figure 3 has lighter voxels around the farther edge, with darker voxels inside, resembling the corner of a hollow object. This is the basis of a more complex hierarchy that will predict an objects function given its form (i.e. Given an object with form β arranged in pattern γ, the probability of being capable of performing function δ is ε). To calculate the output of a given layer in the convolution, Equation 2 is used.

$$Cl_n(x) = -(k * x) + b \tag{2}$$

where x is the input, k is the learned kernel, $*$ is the convolution function and b is the learned bias. An example of this can be seen in Figure 4. Note that this figure demonstrates the output of this layer for just one of the kernels learned. Thus a layer containing n kernels will contain n such outputs. Also note that the output size of these layers is roughly the same size and dimensionality of the input. For example, for an input of size $32 \times 32 \times 32$ to a layer of 16 kernels, stride 2, and kernel size $5 \times 5 \times 5$ will yield an output of $16 \times 15 \times 15 \times 15$. The first parameter is given by the number of kernels (16), while the other dimensions are given by $(32-5)/2+1 = 15$.

3.2.5 Max Pooling Layers

An important component of Convolutional Neural Networks are Max Pooling layers. These layers serve several purposes, including reducing the memory needs of a network. This allows a faster training process, and helps control overfitting. In certain architectures this is an important part of building a more high-level model as well, as discussed in Section 3.3. An example of a pooling operation can be seen in Figure 4 on the right hand side. The Max pooling operation will consider a region of an input and will output the maximum value in that region. Note that the pooling operation output has a much smaller size.

3.2.6 Dense Layers

Dense layers, (or fully connected layers) are a classic Artificial Neural Network structure. In these layers, each neuron is connected to every output from the previous layer, each with a distinct weight. For example, if the previous layer has 1000 neurons, a dense layer of size 50 will contain 50 sets of 1,000 weights, or 50,000 parameters in total, similar to 50 instances of the neuron in Figure 2, each with $n = 1000$ inputs. Each network design contains two dense layers as the final two layers. The first is meant to reduce the size of the data, while the second is meant to output a vector of regressed values for each of the Functional Quantities. These layers are a common element in deep learning when a concise and robust representation of the data is needed, as they give a smaller dimensional vector whose elements are a function of the entire input.

3.3 Neural Network Design

This work proposes a novel network design, designed to be deeper than existing classification networks. Convolutions are a critical part of image analysis because of their ability to capture local resemblances using learned filters. At the bottom layer, the model operates on the raw input, in this case, a binary voxel grid. This voxel grid is analyzed by a set of kernels, which represent learned patterns. In the 2-Dimensional space, these patterns might be "stripes" or "blue". Similarly, in the 3-Dimensional space, these patterns may be "corner", "vertical plane", or "horizontal plane". Subsequent layers can become much more difficult to elucidate, which will be discussed in Section 5.1.4, but they will encompass combinations or patterns of these learned kernels, such as repetition. This deep design will instead convolve and pool once again these pooled layers before the dense layers. By using the ability of convolutions to contextualize low level patterns, the deep design learns higher level designs at the expense of generalizability, which can be regained by data augmentation, discussed further in Section 3.5. This is accomplished by stacking layers so that each feature can represent a larger region of space. For example, two stacked layers of $3 \times 3 \times 3$ layers mean that one value in this output, can use as input every value from a $5 \times 5 \times 5$ voxel range (known as the effective receptive field of this unit), and the pooling process will double this range again. This also requires fewer parameters ($2 \times 3 \times 3 \times 3 = 54$ while $5 \times 5 \times 5 = 125$). By adding a second set of this Conv-Conv-Pool architecture, with a larger number of filters, the authors hypothesize that a more discriminative feature map of the entire object is created by spanning a larger receptive field. The validity of this hypothesis is tested in Section 5. The additional non-linearity provided (both from the activation function LeakyReLU, and the pooling operation), allows these feature maps to be both more expressive as well as spatially expansive by covering the entire input. This second Conv-Conv-Pool unit also contains a larger number of kernels, which will learn patterns of the smaller number of basic kernels. It is common to increase the number of kernels for deeper layers so that the number of feature maps increases as the size decreases. Additionally, dropout layers are employed after each of the convolution-convolution pool units, as well as between the dense layers. This will prevent overfitting by randomly changing the output of each major unit of the network even given the same input. This network also employs Batch Normalization after each convolution which normalizes the activation of each layer of the network. Finally this network concludes with fully connected layers of size 1024 and p neurons where p is the number of functional quantities and depends on the variation of the network. This is discussed further in Section 5.1.1. The size of this first dense layer is selected to keep the number of parameters low, which makes the training process faster, more

efficient, and prevents overfitting.

Because of the difficult nature of this problem (which can map to continuous values in a relatively small range) care must be taken, especially with the later layers of this network. Generally, a linear activation is employed for regression problems, but this presupposes certain characteristics of the data. For example, in this data, there is a clear difference between a functional quantity of 0 and a functional quantity of > 0, namely this is the difference between an object performing a function and not, so an activation function might be appropriate.

Part of this work involves investigating which variation of the architecture shown in Table 1 is best suited for solving the difficult problem of mapping volumetric input to a real valued output. The primary difference between the three variations in this work comes in the final layer. First, the ability of this architecture to make a binary classification is tested. To do this, a *tanh* activation function is employed. This will squash the output to between -1 and 1, which was shown in Glorot and Bengio to be more effective than *sigmoid* (which squashes input to between 0 and 1) for most problems [42]. This is due to the behavior of the backpropogation process and the slow nature of training when inputs are saturated at 0, in addition to the asymmetric nature of this distribution. Since this is a qualitative problem in nature (a simple binary classification), the performance is expected to be better than the more complex regression problem. The second variation uses no activation function, and the network is trained to learn the target values.

The third variation proposed involves making a pseudo-classification over several possible values for the quantity. This is the preferred approach for most regression problems with neural networks since absolute loss can be difficult to quantify, and the usefulness of the loss gradients are limited. Furthermore, as outlined above, the target values of this approach are non-linear in nature, so modeling them as a linear regression target can introduce imprecision in the data. It may be that the nature of this problem makes it best suited for a cascade type problem (for example, a binary classifier chained to a linear regressor, if the binary classifier returns positive). However, this is outside the scope of this work, and indeed jeopardizes the generalizability of the model. Instead, this variation proposes mapping the output to a set of p softmax vectors, which each contain a binary vector corresponding to which binned Functional Quantity the input belongs. For example, the fastest objects would yield a vector of $\langle 0,0,0,0,0,0,0,0,0,0,1\rangle$ during training, since the value falls in the final bin and hence, has a numerical value of 1. Additionally, the final layer of this model uses a *softmax* activation function, which squashes the output to the range (0,1) and ensures that the resulting vector adds up to 1. This allows the output to be modeled as a distribution and an expected value can be calculated over this distribution using the discrete values. This has been shown to work in other regression contexts, such as age regression [43]. This is also beneficial since the model can output a confidence value for each estimation in addition to the value itself. Finally, this simplifies training by allowing more variation in final layer output, since the difference in gradient is emphasized over the raw value of the output, due to the softmax activation. Other base network designs may be able to accomplish this task as well, and the design of this network is contrasted with one such network in Section 5.1.3.

Table 1: The Proposed CNN design. Note for each LeakyReLU, $\alpha = 0.1$.

Type	Neurons	Size	Activation	Purpose/ Benefit
3D Conv	16	3	LReLU	Feature extraction of raw object
3D Conv	16	3	LReLU	Feature extraction; raise receptive field
Max Pool	N/A	2	N/A	Transformation invariance; raise receptive field
Dropout	N/A	0.3	N/A	Regularization
3D Conv	32	3	LReLU	Feature extraction; raise receptive field
3D Conv	32	3	LReLU	Feature extraction; raise receptive field
Max Pool	N/A	2	N/A	Transformation invariance; raise receptive field
Dropout	N/A	0.3	N/A	Regularization
3D Conv	64	3	LReLU	Feature Extraction
Dense	1024	N/A	ReLU	Classification Vector
Dropout	N/A	0.3	N/A	Regularization
Dense	1024	N/A	ReLU	Classification Vector
Dropout	N/A	0.5	N/A	Regularization
See Sec. 3.3	?	N/A	See Sec. 3.3	Classification

3.4 Functional Quantity Targets

This method seeks to map object form, as represented by binary voxel grids, to functional quantities for different functions. These functional quantities are real valued, and are meant to represent both magnitude and rank of these functional quantities on a per class basis. The choice of the values of these functional quantities is important since the resulting model will be evaluated on its ability to correctly classify both magnitude and rank of the input objects. Additionally, other considerations must be taken to avoid other problems such as class imbalance. For instance, the maximum speed among all object classes is *airplane* at 500 MPH, as shown in Figure 5. A target value of 500 for this class however would result in substantial loss in the case of a mistake, which would dominate the loss of other functions. This would lead to any optimizer overfitting for functional quantities whose magnitudes were largest, and especially for classes whose quantities were largest. To combat this, normalization must be performed. However, a simple normalization such as dividing by the maximum value will not suffice either, since this would render low values for high magnitude quantities (such as speed) to be so close to 0 as to become too unimportant. It is critical that the difference between 0 and a low value be pronounced since this answers the question "can this object provide conveyance". Instead, prior to normalization, the functional quantity types whose ranges are too large are converted to a logarithmic scale, which can accurately capture magnitude without losing meaning for very

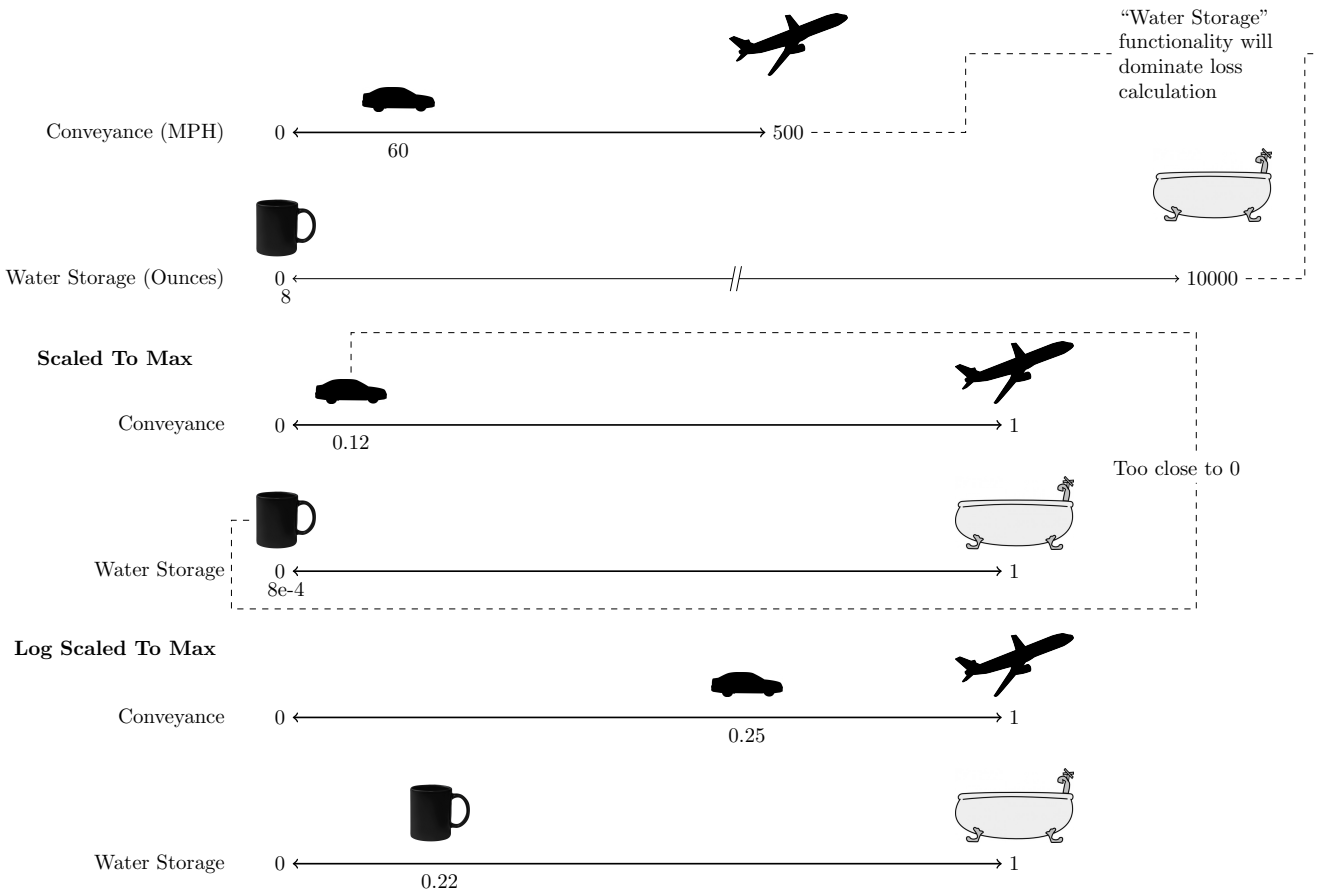


Fig. 5: The scaling scheme of the functional quantity targets

high or very low values. Finally, these values are scaled to be between 0 and 1, by dividing by the log of the largest value. This has the effect of preserving rank and magnitude without overfitting for higher magnitude classes within a functional quantity or higher magnitude functional quantities. This will also preserve the semantically meaningful relationships such as *cars* move while *chairs* do not, and *airplanes* move much faster than *cars*. Because of the sparseness of this dataset, other methods were considered, including a box-cox transformation, and a cube root transformation, both of which are used on datasets with these characteristics. However, empirically the coefficient of determination against a normal distribution was greatest across each function when scaling to log base 10 values.

3.5 CNN Training

To train the CNN, as with any machine learning abstraction, two components are required: Training Input and Training Output. The input is k 3D Voxel Grids of the artifacts described in Section 3.1. The output is a set of k vectors where each vector $c \in C$ is of length p, and each entry corresponds to one of the p functions being considered by this model. For example, if a given model is capable of being sat on, by up to 500 lbs (the maximum allowed by this experiment) but not storing water, or providing conveyance ($p = 3$), the vector c representing this artifacts capabilities, for the regression variation (discussed in Section 5.1.1) would look like $\langle 1,0,0 \rangle$. These capabilities are assigned based on the object class, as defined by the dataset. To avoid training issues, all functional quantities are normalized to the range $[0,1]$. The training data is also

sub-sampled to correct for the class imbalance problem. For each class which performs at least one function, n samples are selected randomly, where n is chosen to be approximately the number of samples in the smallest class. This ensures that the values for each quantity are roughly evenly distributed.

During the training phase, the input is provided to the network, that performs a series of convolutional, pooling and activation operations, finally outputting a vector. Comparing this training output with the desired output yields a loss, and adjustments are back propagated through the neural network towards the goal of minimizing this loss on subsequent iterations. The Training Data is iterated through and the error (Mean Squared Error (MSE) between predicted and Training Output) is propagated through the model for a set number of epochs.

3.6 Network Testing

Finally, this trained model must be evaluated. To evaluate the effectiveness of this method, a leave one out approach is employed. The dataset used in this method contains a number of similar *classes*, that each map to some set of functions. For example, the class *bench*, can be used for *sitting* as can the class *chair*. This is done to prevent overfitting for object appearance, and to evaluate on new novel input, that may look substantially different than anything in the training set. For example, if a user uploads a new artifact to an artifact sharing service, it may perform some function but look very different from any object ever seen before. Thus, it is important to leave out entire classes of objects, so that the model can be evaluated for more than just its ability to map appearance to class and class to function. Then this trained model will be evaluated by providing it with input from the *bench* class, and the model will be evaluated for its ability to correctly predict that all benches can be sat on. Additionally, this demonstrates the capability of these deep networks to perform several recognition tasks simultaneously. In other words, the same trained network will be able to estimate each of these functional quantities in one process, which eliminates the need for separate models for each function, and allows the model to be easily extended for new functions in the future. For evaluation, first this method must be validated, using the training and testing set provided by the dataset. This validation will be performed by training on the training set across all classes, and testing for accuracy on the test set. Next, the leave one out design will be evaluated. For each left out class, all examples of this class will be provided as test input to the trained network. The trained network will output a series of quantities for each object, which will be evaluated for their difference between this and the ground truth data. Particular care must be paid to classes with more than one functional quality, since this will greatly illuminate the ability of this model to simultaneously and accurately calculate these functional quantities. Finally, this model will be evaluated for its ability to understand rank and order of magnitude by comparing the predicted quantities to their nearest ground truth quantity.

4 Case Study

4.1 Functional Quantities

For the case study of this method, five functions are considered: *Sitting*, *Storing Liquid*, *Conveyance*, *Emitting Sound*, and *Displaying Image*. The input data used is drawn from the Princeton ModelNet40 dataset [32] and the Shapenet Dataset [33]. This dataset consists of 25 classes, each of which is coded according to its ability to perform a specific function.

Table 2: Functions, classes and quantities considered by this method.

Function	Classes
Sitting (Pounds)	Bench (500), Chair (350), Stool (300), Sofa (450), Toilet (1000)
Water Storage (Ounces)	Cup (8), Bathtub (1000), Flower Pot (32), Toilet (128), Vase (50), Bottle (20), Bowl (12), Sink (2300)
Conveyance (MPH)	Car (60), Bus (50), Boat (30), Airplane (500), Train (70), Motorcycle (60)
Emits Sound (Decibels)	Radio (100), Laptop (90), Loudspeaker (120), Headphones (100), Telephone (80), Cell Phone (80)
Displays Image (Lumens)	Laptop (500), Monitor (400), Cell Phone (400)

Classes which could not perform a function were left out, as were classes whose function requires external manipulation (for example, a piano needing a human in order to make a noise). Note that since each function is assessed independently, the network can independently reject each function, even though no objects without function were used during testing. The models in this dataset were also augmented by rotating them around the gravity axis 12 times. The specific labels can be seen in Table 2. These functions were selected based on two criteria: substantial model availability in public datasets, and simplicity (i.e. no human interaction is required). As with many learning problems, a lack of training data It is worth noting how function and class interact. Each class (e.g. chair, bench) carries with it a function (e.g. sitting) and a quantity (500 lbs). In this experiment, for each other function considered (e.g. displays image), that object carries a quantity of 0, meaning it cannot perform that function.

One challenge not considered by this method is that of object scale. Since all models are scaled to the same size along their longest edge, object scale is impossible to infer from the raw data alone. Future work may consider a modification of this network to accept metadata (such as scale or material), but this is considered outside the scope of these experiments. As the models are anchored along the gravity axis it is possible that scale is still implied, if not explicitly stated. For example, larger objects in this dataset tend to be wider than they are tall (e.g. bathtub, train), while smaller objects are often taller than they are wide (e.g. water bottle, cup). First, each object was converted from their native format to a voxel grid [44][3]. Due in part to memory constraints, each voxel grid was downsampled to a grid of $32 \times 32 \times 32$ voxels. Subsequent work by Riegler has enabled larger resolutions to be used, however, their implementation has not been made available [45]. The Network as specified in Section 3, was implemented in Python using Keras [46], that is based on the Theano Library [47].

The baseline models were trained for 200 epochs, and the left out models were trained for 50 epochs. Additionally, to validate that these forms are being evaluated for functionality, a classwise leave one out strategy is employed. In other words, for a given class among the c classes that possesses functionality p, the remaining $c-1$ classes are used in the

[3] http://www.patrickmin.com/binvox/

training set. During testing, a test set of the $c-1$ classes in addition to the entirety of the left out class are evaluated for the network's ability to recognize functionality not just among seen classes, but among unseen classes as well. For example, the *bus* network will be tested for its ability to determine that buses can be used for conveyance, despite never having seen a bus before.

These networks will be evaluated on several metrics. First, their ability to correctly recognize similar input will be used to select which variation merits further examination. Next, the networks will be evaluated for their ability to accurately predict the functional quantities of dissimilar input, both including and not including their ability to correctly identify that an object cannot perform a function using an argmax of predictions. These left out networks will also be evaluated for their ability to correctly predict rank and order of magnitude of these functional quantities. Their expected values will also be examined, as the goal of this work is not just a correct classification, but an estimation of a real world value. Due to the nature of the data, it will not always be possible for argmax predictions to be accurate, so the rank within the function must also be examined using an accuracy score calculated as follows:

$$FQ_{acc} = \frac{P_> + P_<}{T_> + T_<} \tag{3}$$

where $P_>$ and $P_<$ are the number of true functional quantities which are greater than and less than the predicted quantity, respectively, and $T_>$ and $T_<$ are the number of true functional quantities which are greater than and less than the true quantity. In other words, the toilet network may not be able to tell exactly how much water a toilet can hold, but it will still be evaluated for its ability to determine that a toilet can hold more than a cup and less than a bathtub. This relative accuracy with respect to other quantities within the function will provide more insight into the networks ability to predict novel values on novel data.

4.2 Separating Function and Aesthetics

As users download artifacts from open source repositories, there may be a tendency to misinterpret the intent of an artifact designed for decoration. This is a complex problem; decorative items are intentionally made to resemble their functional counterparts, but often lack key elements which ensure that they function as intended. While Section 4.1 assumes that all objects of a class perform a function equally well, this section considers head-wear artifacts whose purposes are mixed: Hats/Caps and Helmets. Head injuries resulted in over 65,000 cases of missed work in 2012[4], so the ability of this model to determine if a given artifact will keep a user safe is key. The model is asked to predict a variable indicating if the head-wear was designed with safety in mind (in contrast to decorative items). To do this, each helmet was manually tagged to indicate if the given helmet is decorative (that is, if the designer's intent was to create a helmet that could be used for protection). In the dataset, functional artifacts were primarily drawn from racing, military, or sports contexts, while the decorative ones were rooted in works of fiction such as video games or film (see Figure 12). Additionally, all instances of

[4]http://www.nsc.org/learn/safety-knowledge/Pages/injury-facts.aspx

hats were also included in this experiment, none of which are capable of providing protection to the users head. In total, there are 218 artifacts in this experiment (including 160 decorative and 58 functional). These models are trained for 20 epochs using 5 fold cross-validation, and are evaluated for precision and recall, as well as accuracy.

5 Results and Discussion

This section presents the results of this work, and evaluate how accurately the functional quantities are predicted.

5.1 Functional Quantities

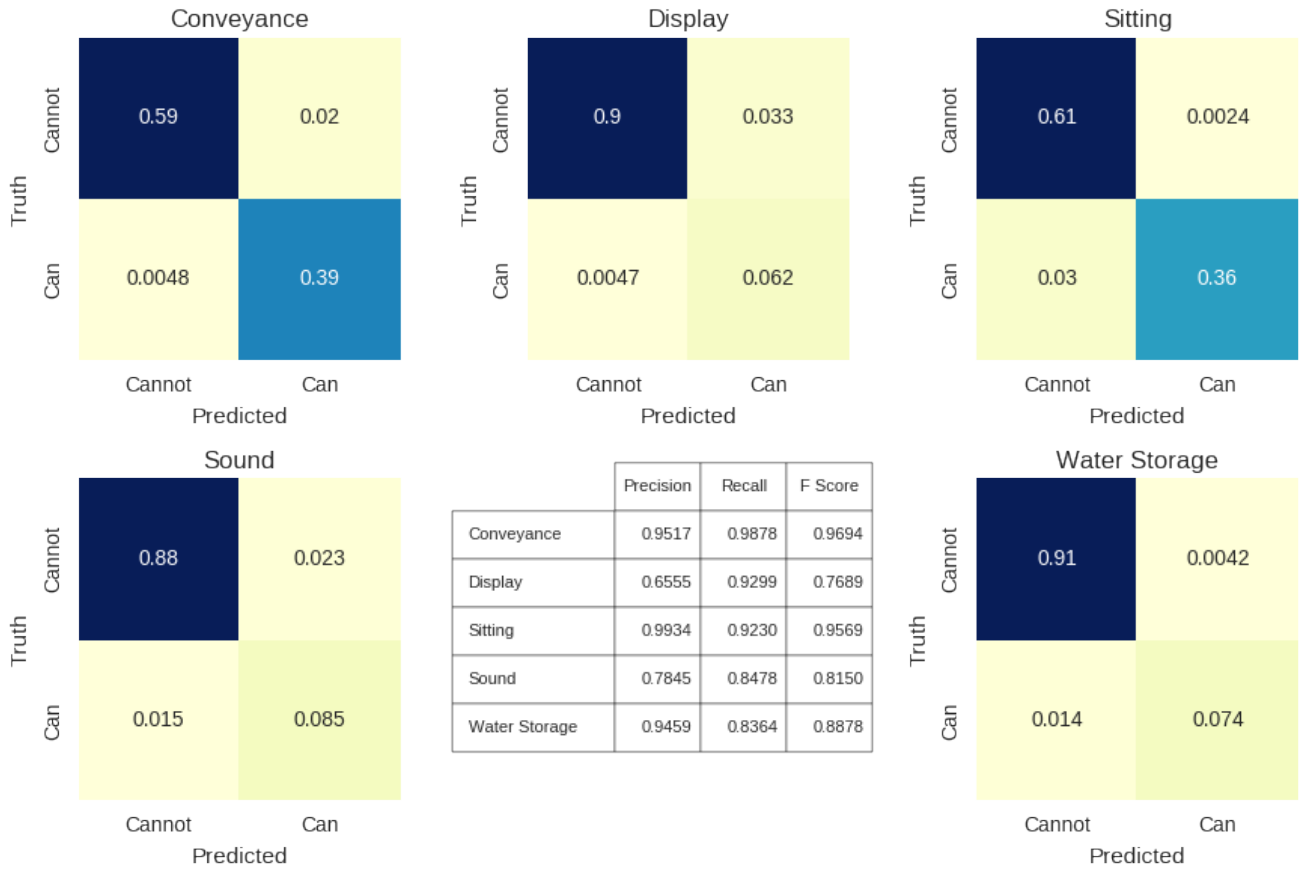
5.1.1 Full Dataset Baseline

Several evaluation methods are proposed in this section. First, the basic network outlined in Section 3 is trained with one exception: the final layer is replaced with a simple Dense layer with 5 outputs, and a tanh activation function. This is meant to demonstrate that the network can make qualitative assessments of object function. Briefly, this network will output a value in [-1,1], where -1 indicates that the object cannot perform the function and a 1 indicates that it can. Since the network outputs predictions on a continuous scale, a binarization scheme was performed which rounded values > 0 to 1, and all others to -1. This network is trained for 200 epochs using the Adam optimizer [48], and tested with the given testing set for the 25 classes. The results can be seen in Figure 6. As can be seen for each function, this solution is well suited to make quantitative assessments on novel but similar input (e.g. this model is trained on one set of cups and tested on another set of cups). Among the 5 functions tested, the conveyance function performed best, with an F Score of 0.9694. Similarly, Sitting yielded an F Score of 0.9569, and performed nearly as well. All 5 classes yielded an F Score above 0.75, the lowest belonging to Display, whose precision was low, possibly due to a low number of true positives (just 6.7% of all samples). In fact, moving the binarization threshold from 0 to 0.6 increases the F Score to 0.8411, indicating that generally the false positives score lower on this scale than the true positives.

On a per class basis, every object performed at above 90% accuracy with three exceptions. The worst performer was Telephone and the Display function (13.0%), which may be due to Cell Phone models being included in the telephone dataset, causing low performance in that test. The other two are Loudspeaker/Conveyance (84.6%) and Bathtub/Sound (82.5%), emphasizing the difficulty of this task.

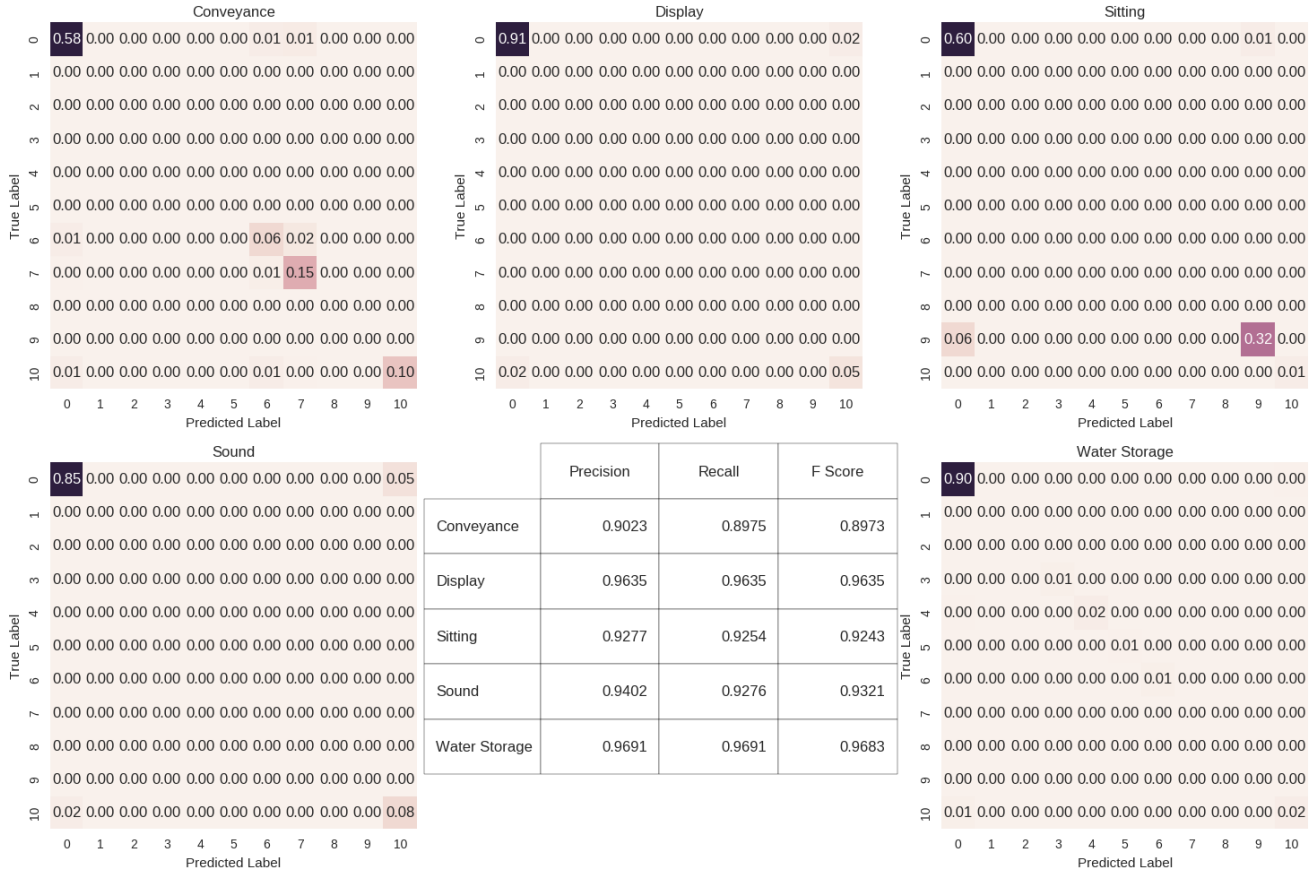
Next, the same network is tested using a multi-class classification scheme. In this model, rather than outputting five quantities, one for each function, the network will output five softmax vectors, indicating to which binned quantity the model believes the object belongs. Additionally, since a softmax label can be treated as a distribution, the final expected value of the functional quantity can be calculated for Function f_i according to Equation 4:

$$FQ_i = \sum_{t \in d} O_{i,t} \times FQ_{i,t} \tag{4}$$



	Precision	Recall	F Score
Conveyance	0.9517	0.9878	0.9694
Display	0.6555	0.9299	0.7689
Sitting	0.9934	0.9230	0.9569
Sound	0.7845	0.8478	0.8150
Water Storage	0.9459	0.8364	0.8878

Fig. 6: Confusion Matrix for the binary (qualitative) variation of this network



	Precision	Recall	F Score
Conveyance	0.9023	0.8975	0.8973
Display	0.9635	0.9635	0.9635
Sitting	0.9277	0.9254	0.9243
Sound	0.9402	0.9276	0.9321
Water Storage	0.9691	0.9691	0.9683

Fig. 7: Confusion Matrix for the softmax regression variation of this network

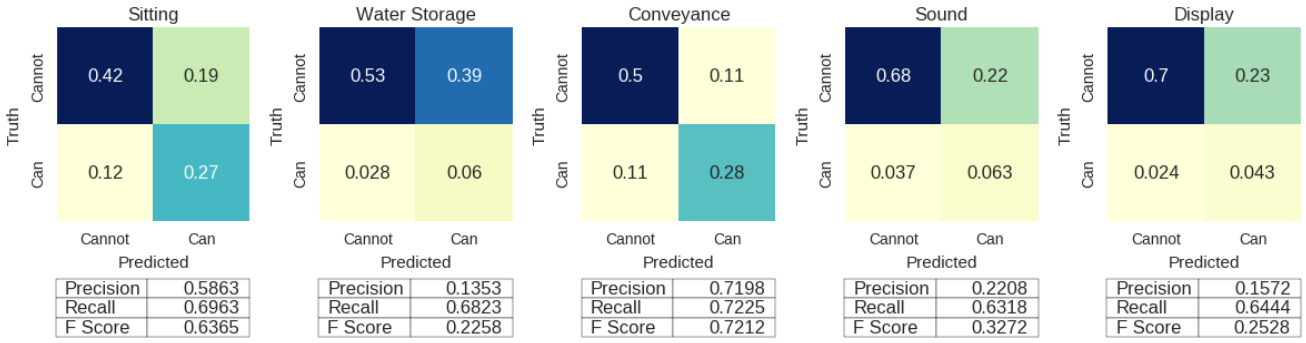


Fig. 8: Confusion Matrix for the absolute regression variation of this network

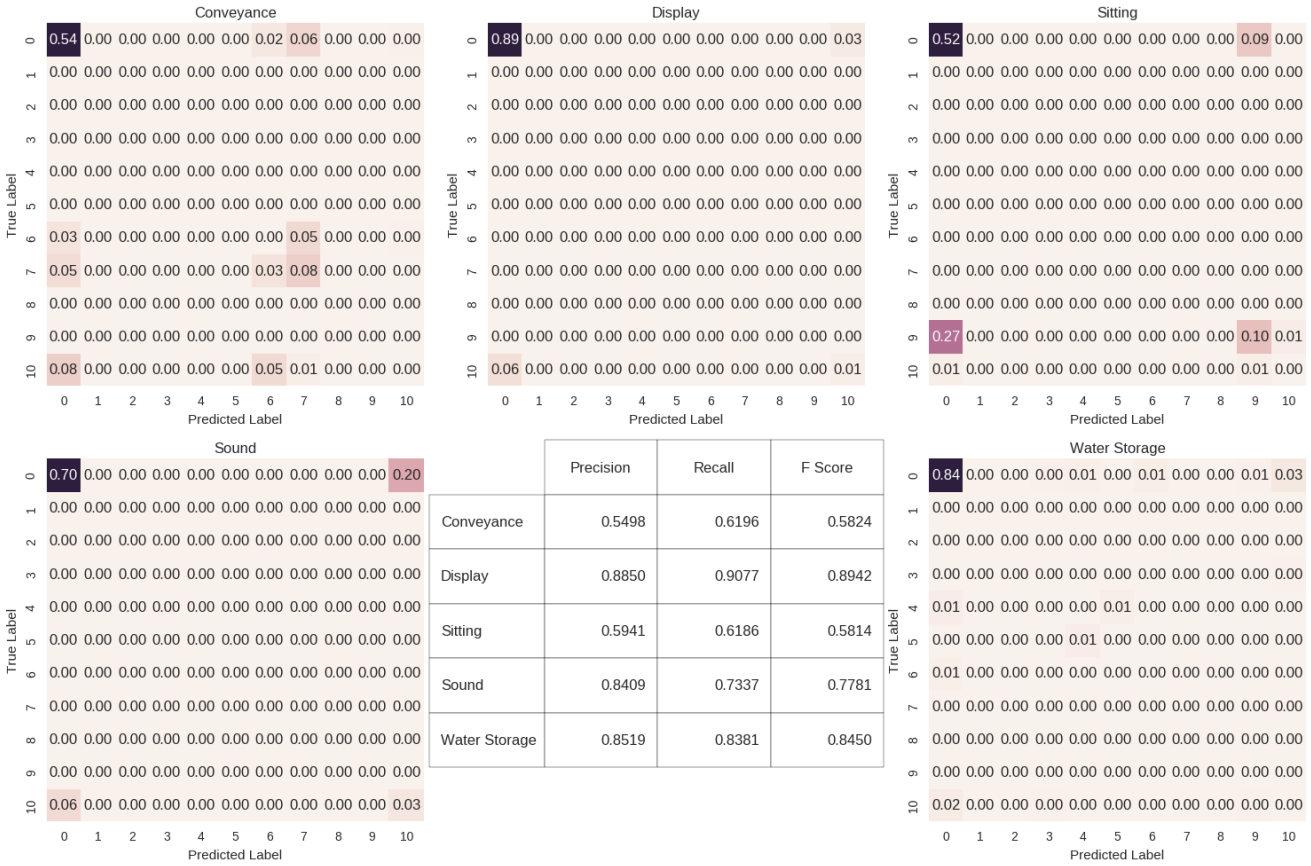
where d is the number of types for a given function (for example, 6 for sitting), O_t is the output of the trained network for type t, and FQ_t is the quantity associated with type t (for example, 500 lbs for bench). This will be used to evaluate the difference with respect to ground truth.

The results of this classification can be seen in Figure 7. To calculate this confusion matrix, the argmax of the softmax vector output was taken, and this was compared to ground truth. Note that the lowest F Score for each of the 5 functions is 0.8973, and that for 3 of the 5 functions, the F Score is higher for this variation than for the qualitative version. This may be due to overfitting during training as a way of combating the class imbalance problem, which may be detrimental to the performance of the qualitative problem. This may also indicate that this problem is especially facilitated by this type of softmax logistic regression. It is also worth noting that for many of the classes, there are some ground truth labels which are never proposed, meaning that for some functions (Display and Sound), there is little difference between this variation and the binary variation, making it notable that these functions performed better under this constraint.

Finally, a straightforward regression approach is employed, where the final layer is a simple output whose goal is to find the real value of the objects functional quantity. The results were binarized using the following binarization process:

$$y' = \begin{cases} 0 & y \leq avg(y) \\ 1 & y > avg(y) \end{cases} \tag{5}$$

where y' is the binarized value, y is the predicted value, and $avg(y)$ is the average predicted value across all samples. A per-function binarization scheme was necessary because of the different distributions across the functions tested. This network yielded low F Scores including below 0.5 in 3 of the 5 functions (Sitting: 0.6365, Water Storage 0.2258, Conveyance: 0.7212, Sound: 0.3272, Display: 0.2528). Since this design required an unusual binarization scheme (many of the functions had a hard maximum value less than 1), and performed less well than the softmax approach, this variation of the network was not used going forward.



	Precision	Recall	F Score
Conveyance	0.5498	0.6196	0.5824
Display	0.8850	0.9077	0.8942
Sitting	0.5941	0.6186	0.5814
Sound	0.8409	0.7337	0.7781
Water Storage	0.8519	0.8381	0.8450

Fig. 9: Confusion matrix for the aggregated left out classes

5.1.2 Left Out Classification Accuracy

Since the categorical variation of this network performed better than the absolute regression, it is selected and evaluated using a left out classification scheme. The confusion matrix for each of the five functions can be seen in Figure 9. This is the aggregate of only left out classes across each of the 25 training runs, argmaxed, and compared to ground truth. Naturally, the accuracy is much lower for each of these functions, since each of these trained networks has never been trained on input resembling this test input before. So, in contrast to networks trained on 25 classes and tested on 25 classes, this measures networks trained on 24 classes and tested on 1 class. This validates that the model learns latent variables which can be applied to new input. Generally speaking, most classes have similarities with other classes which share a function, and this section tests the ability of a neural network to learn these similarities in latent variables (see research question ii).

Nevertheless, this model still performs well on these functions. The lowest F Score calculated (under a weighted scheme) is 0.5814, significantly better than chance. However, these F scores are kept artificially high by their ability to correctly reject novel input, so deeper examination is merited. Table 3 shows the accuracy for each left out class, for each function it can perform, under argmax binarization. Some notable high performers include cellphone/sound, telephone/sound, train/conveyance, bus/conveyance, laptop/display, and bench/sitting. However, argmax binarization is not particularly well suited for this task. For example, consider the airplane, which is the fastest item in the dataset. This network would be required to infer that the airplane is faster than anything seen before in order to obtain correct output.

Thus, the binarization constraints are relaxed in the column labeled Relaxed Accuracy. This gives a more accurate picture of if the model believes the object is capable of performing this function at all. This yields much better results. For example, the class Bowl sees a jump in accuracy of 84% Flower Pot increases 81%, sink increases 66% and vase increases from 0 to nearly perfect 96%. The Water Storage function by far benefits the most from this relaxed binarization method (increasing on average 62%), although conveyance also sees a substantial jump (30% on average). Airplane, car, motorcycle and boat each perform much better on this more relaxed binarization. The toilet class also saw a substantial increase in the sitting function indicating that the network had ascertained that a toilet could be sat on, but was unable to determine how much weight it could support. The "Can Or Cannot Accuracy" is a variation of this binarization in which the non-zero likelihoods are summed, and this is used to determine if the network believes if the object can perform a function or not. This is slightly different than argmax and can be more useful in the case of several non-zero estimations "splitting the vote" allowing the "can not" estimation to dominate.

These results may also be a function of lacking a full distribution of target values across each of the bins. Indeed, as is clear from the confusion matrices, none of the functions utilize all 11 of the binned values, thus ensuring that their weights remain 0 and will never be the direct output of an estimation. Put another way, this network will not be able to output an estimation of 0.3 for an object if it has never seen an object whose functional quantity is 0.3 before. This becomes particularly clear under leave one out conditions, since the functional quantity distributions can become even more sparse. This can be seen in Table 3 for classes such as Boat, Sink, and Bathtub. A more fully featured dataset may be able to overcome this, but that is outside the scope of this work. Nevertheless, the expected values for this work can also be calculated, and will provide a more complete picture of the networks belief of these objects capabilities.

The FQ_{acc} is a quantitative score which measures broadly the models ability to not only predict an objects intended use, but a measurement of quantity as well. Put simply, any model which believes a mug and a bathtub can be used in the same way is an incomplete model, so it is important to measure that the model understands the ways in which a mug and a bathtub are the same and the ways in which they are different (that is, the volume of water being held). The same can be said of many other classes in this experiment, and this score yields results which are encouraging. To be clear, the output values are not pure regressed values, but rather a prediction of where along a scale (as presented by the training set) the network predicts an object lies. Since this method makes a distinction between FQs of 0 (which are counted in the first bin and indicate that the object cannot perform the function at all) and very small FQs (which are counted in the second bin and indicate a belief that the object can perform the function, but in a small amount relative to the maximum), care must be taken when re-binning the calculated expected value. For the results shown in Table 3, all values below 10^{-2} were placed in the first bin (meaning the object cannot perform this function), and all values otherwise were rebinned using the same ranges as during training.

While some of these results can be attributed to the sparse layout of the various classes, as is expected when using a metric which bases its accuracy on relative values rather than actual values, these results help emphasize the effectiveness of using convolutional neural networks on volumetric data. For many classes, the FQ_{acc} is higher even than the relaxed accuracy. This is especially clear in classes whose functional quantity space is diverse, indicating that this approach may be appropriate for larger datasets going forward. This gives a useful measurement of objects whose argmax may indicate that

they cannot perform a function but whose distribution describes a strong belief that they can. The most notable performances in this metric are Cell Phone, Headphones and Sofa. For each of these classes of object, despite having never seen this type of object before, networks trained on other items were able to determine that headphones and cell phones can make sound, cell phones can display images and sofas can be sat on. Other sitting objects performed well as well, with stool, chair and bench each with $FQ_{acc} > 0.85$, consistent with the finding that sitting as a function yielded the highest average FQ_{acc} for left out objects at 0.7932.

The difference between performance across functions can serve as an approximation of how well the appearance of an object corresponds with that object's ability to perform a function. With this in mind, the five lowest performing class/function pairs were either Display or Sound. This illustrates one of the limitations of this work, which is that without rich visual information, and a degree of commonality of appearance across a function, the model is not able to provide accurate predictions. For instance, monitors are fairly homogeneous in appearance, but give very little indication that they can be used for display, and this is reflected by the models very low accuracy for this class. In the case of a large online repository, where a more diverse dataset is not only expected but a critical part of the process, a more robust model can be built based on a fuller set of functional quantities.

5.1.3 Baseline Neural Approach

This network will be tested against the VoxNet design proposed by Maturana and Scherer [34]. This network will be modified slightly to output functional quantity estimates instead of classification. The VoxNet Network design uses fewer layers with more parameters to predict the functionality of a given object. This network was chosen because it is known to work on a similar, but distinct, problem, and can therefore serve as a baseline. The key difference between VoxNet and the network proposed in this paper are i) VoxNet uses just 2 convolution operations ii) VoxNet does not batch normalize, which slows down training and prevents regularization iii) VoxNet is designed for a single object classification task. To address iii, the network was altered to output 5 softmax vectors, similar to the variation which performed best for this problem described above. Additionally, an Adam optimizer was used, so that the same optimization method was used for both architectures. It is worth noting that during training this model was not stable over 200 epochs, so the model which performed best on the training data was selected and used for these results. This is in contrast to the deep model proposed which saw loss that fell reliably.

The results can be seen for the 25 class problem in Figure 10. Keeping in mind that this model was selected to perform best on the training data (the final form was predicting all "cannot perform this function"), the model performs comparably to the proposed model in this paper. Notably, this shallower model performed better on Conveyance objects (F Score of 0.9107 to 0.8973) and not as well on Sitting objects (F Score of 0.8956 to 0.9243). These differences may be related more to the nature of these functions, rather than the model architecture itself.

To explain this, consider the relative depths of these two networks. VoxNet contains just one set of two convolutions and a max pooling, whereas the new network design contains more layers, and thus more convolutions, it does also have fewer filters at the lowest level (32 for VoxNet vs. 16), which operates directly on the input. Since there is a spatial and

Table 3: Accuracy of each left out class and function. For brevity, W = Water Storage, Si = Sitting, So = Sound, D = Display and C = Conveyance

Class	Func	Argmax Acc	Relaxed Acc	Can/Cannot Acc	FQ_{acc}
Vase	W	0%	96.3%	96.4%	81.3%
Bowl	W	0.6%	84.2%	84.9%	49.2%
Flower Pot	W	1.38%	82.6%	83.9%	79.9%
Cup	W	1.9%	79.6%	81.9%	77.4%
Telephone	So	81.8%	81.8%	81.8%	98.8%
Car	C	45.4%	71.8%	78.3%	69.0%
Motorcycle	C	33.9%	63.8%	77.3%	67.3%
Cell Phone	So	76.0%	76.0%	76.0%	99.4%
Train	C	51.6%	67.2%	73.4%	73.7%
Bus	C	68.7%	69.9%	71.4%	73.9%
Sink	W	0%	66.4%	68.7%	57.7%
Boat	C	0%	66.1%	67.8%	83.6%
Toilet	Si	0%	50.9%	50.9%	40.6%
Headphones	So	50.2%	50.2%	50.2%	99.1%
Laptop	D	47.3%	47.3%	47.3%	85.9%
Bottle	W	6.9%	46.9%	47.2%	77.2%
Airplane	C	0%	42.9%	46.8%	31.7%
Bench	Si	43.7%	43.7%	43.9%	89.3%
Stool	Si	22.7%	33.1%	36.3%	86.2%
Toilet	W	0%	25.4%	27.9%	54.5%
Bathtub	W	0%	27.4%	27.8%	18.3%
Sofa	Si	27.3%	27.4%	27.5%	92.2%
Chair	Si	20.8%	25.9%	26.5%	88.4%
Loudspeaker	So	18.4%	18.4%	18.4%	65.0%
Monitor	D	9.70%	9.70%	9.70%	65.8%
Laptop	So	1.73%	1.73%	1.76%	79.0%
Cell Phone	D	1.11%	1.11%	1.11%	73.9%
Radio	So	0.74%	0.74%	0.87%	21.4%

volumetric loss for each convolution and pooling operation, and these operations are spatially connected, further convolution and pooling means the final prediction is made based on higher level features, rather than lower level ones. Furthermore, there is a spatial and volumetric loss for each convolution and pooling operation, meaning that with deeper networks, more features are considered in context. Typically this will result in an increase in accuracy, but in some cases these low level features can have a higher predictive power. It may be the case here that these low level features better predict the Conveyance

	Precision	Recall	F Score
Conveyance	0.9171	0.9103	0.9107
Display	0.9677	0.9695	0.9680
Sitting	0.9086	0.8988	0.8956
Sound	0.9427	0.9341	0.9373
Water Storage	0.9662	0.9656	0.9654

Fig. 10: Performance on this task by VoxNet Network [34]

class, and further convolution and pooling causes a loss of information, rather than a gain. Another key difference is the size of the filter on the lowest level. The network preserved in this work employs a $3 \times 3 \times 3$ network, while VoxNet uses a $5 \times 5 \times 5$ filter. As discussed, the addition of subsequent fields can raise the effective receptive area past $5 \times 5 \times 5$, but the value of learning specific larger filters may be greater for specific edge cases, such as long flat surfaces. Lastly, while VoxNet has fewer parameters than the model proposed in this work, the larger input layer greatly slows down the training and testing process, meaning that this new model runs more efficiently, while achieving similar results. The larger parameter count also means that this model has the ability to be generalized much more effectively, and could serve as a basis for fine tuning on larger datasets and function classes.

5.1.4 Latent Features

Latent Features are features found in the inner workings of the model that can reveal additional information about how the model works, and what it has discovered. This section discusses what these features mean for object form analysis as it relates to functional quantities. Since neural networks have so many parameters, examination of their parameters, especially beyond the first layer, is neither recommended or nor helpful. Instead, it is more illustrative to examine their activations for specific inputs. Since the activations are spatially arranged, these outputs at intermediate layers can be viewed as filtered images of the input.

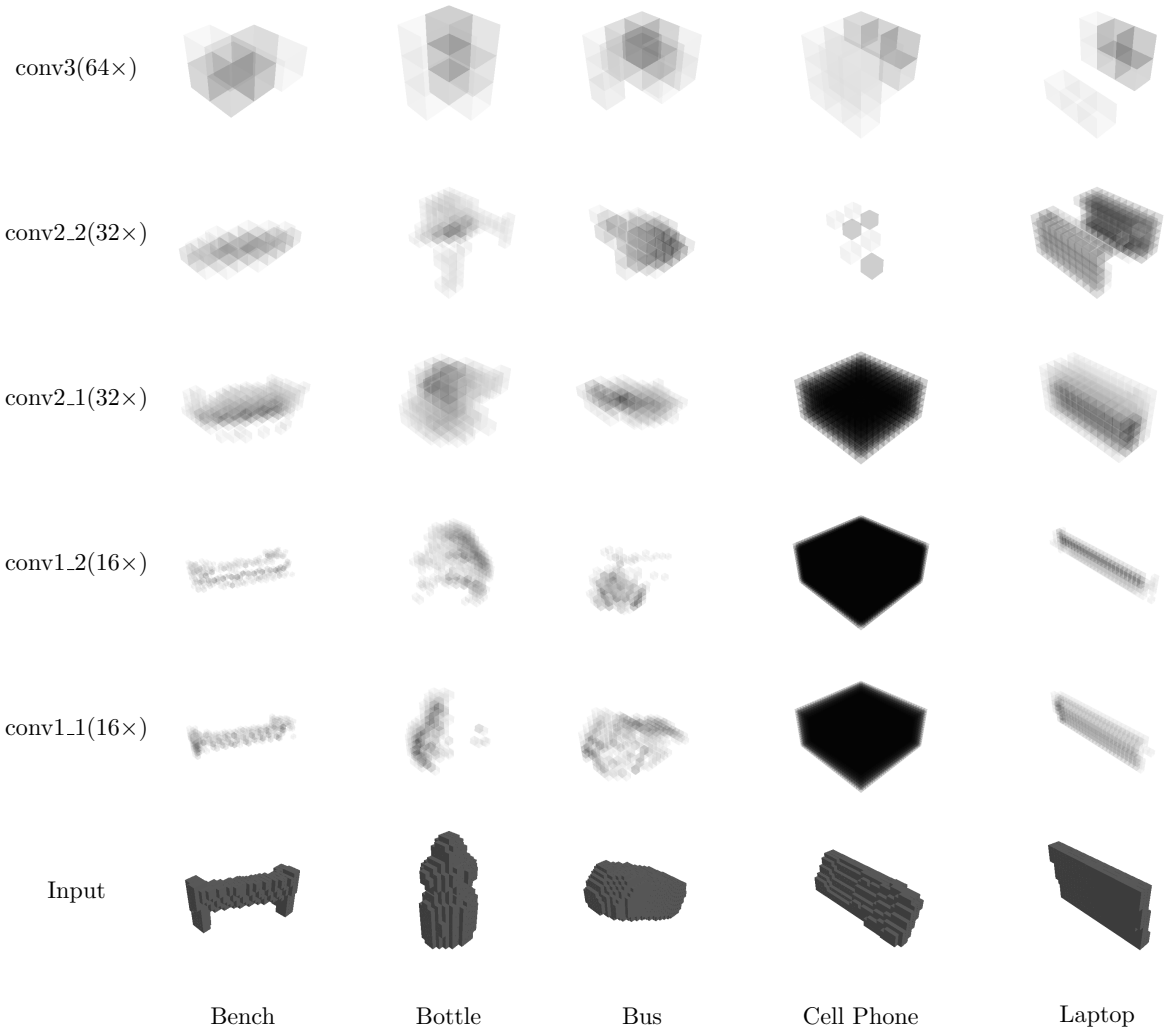


Fig. 11: Activations of the left-out networks on novel inputs for well performing inputs.

To view these latent features, good performers from each of the 5 functional quantities during the leave one out phase were selected. In other words, objects were selected for which there was a great amount of functional certainly despite not having seen that object class before. Each of these objects yielded a correct prediction with high certainty. The best performing examples were Bench for Sitting, Cell Phone for Emitting Sound, Bus for Conveyance, Laptop for Displaying Image and Bottle for Water Storage. The bottom of Figure 11 shows the models. Recall that these models have been voxelized to a resolution of $32 \times 32 \times 32$ losing much of their detail. Even without the majority of the detail of these objects, the network was still able to correctly infer the functional quantities for these inputs.

Above these objects in Figure 11, selected activations can be seen. Recall that the activations of these layers are 4 dimensional (number of filters $\times$ activation space), for brevity and clarity, only activations which contain insights are presented. For clarity these activations have been transsected along the z-axis (except for the conv3 activations) and activations which are below 0 are omitted. Generally, the activations become more difficult to interpret visually the deeper the layer. However, for several of the inputs, the activations are very clear. The activation shown for layer conv1_1 for the bench appears to activate well for inner surfaces of the bench itself, while for bottle and bus, the filter activates well for outer edges,

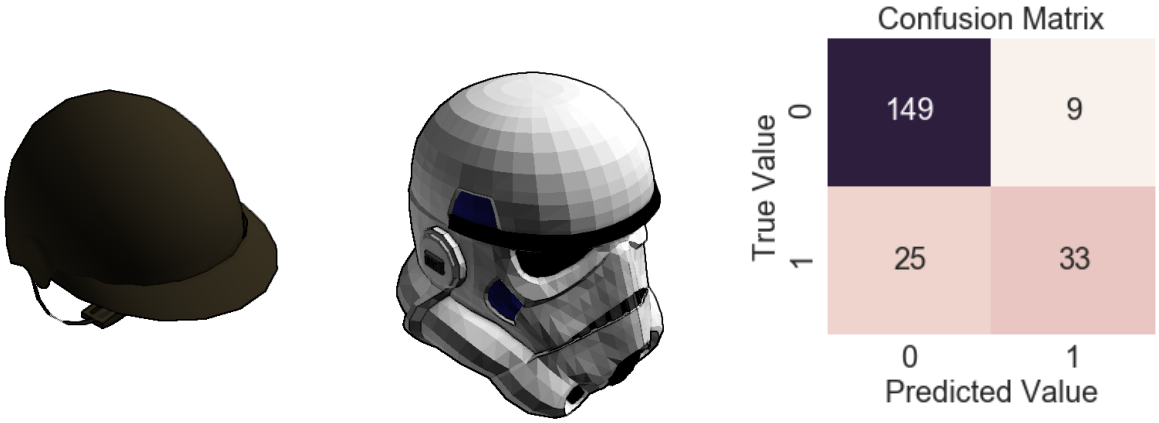


Fig. 12: Headwear from the dataset, the left helmet is functional, while center is purely decorative. Right depicts the confusion matrix of this experiment

particularly upper in the case of the bus. Laptop shows the front edge of the object, while cell phone seems to activate well for non-resident voxels. While this may appear to not contain any information, the activations in conv2_2 and conv3 of cell phone show a clear activation, meaning that while the information does not visualize well, the neural network resolves this prior to prediction. Conv2_2 of the laptop form is included to illustrate that a filter may learn to activate for empty space, and how that can still be used to make a prediction. These figures are generated by a method similar to that introduced by Zeiler and Fergus [49], which searches for highly activated filters. However, their method resulted in visuals which were less easily understood in the 3D space.

5.2 Separating Function and Aesthetics

The purpose of this experiment is to predict if a given piece of head-wear is functional or decorative in design. Across the 5 folds used in cross validation, the average accuracy was 84.4% on test data, with a 0.786 precision and 0.569 recall (see Figure 12). While there is room for improvement in these results, a high precision means that any artifacts which are classified as functional have a high likelihood of being able to provide protection. The fact that a false negative is more likely than a false positive validates the utility of such a method in real world settings. In the instance of safety, a false positive could potentially lead to a user wearing unsafe head-wear in a dangerous environment, while a false negative would only result in an erroneous recommendation to avoid a given helmet. As with the above experiments, this analysis could be easily integrated into an online repository and used to provide recommendations about intended use of a given object. The ability to differentiate between decorative and functional artifacts can also help address some of the limitations of the above experiments by answering more fine grained questions about items within a class.

6 Conclusions And Future Work

This work presented a method which uses deep learning to predict the function of an object based on a provided object form. The ability of these forms to predict functionality is evaluated. While it is true that not all forms are directly related to function, the ability of this method to elucidate which functions are form dependent has many implications. This method is

evaluated against seen and unseen classes of objects to validate that the model is performing as expected.

One of the primary contributions of this work is proposing a new deeper network design, which is deeper than other designs proposed in this space. The performances of this design on the task of form-function prediction sheds valuable information on how such relationships can be established. As shown, the deeper, more non-linear design performs much better at this task than the a baseline shallower network. This indicates that for the process of function prediction, simply viewing an object as a combination of small pieces is not as productive as learning how these pieces relate to one another. These relationships are so important, that a model which has been trained on these relationships can identify not just what an artifact is used for, but to what degree it can perform this function despite having never seen that type of object before.

One major limitation of this work is the selection of functional quantities. While these quantities were chosen to reflect the rank and magnitude of the objects in question, their assignment based only on class introduces an overly simplistic view of how these quantities can vary within a class. Future work may take advantage of the generative capabilities of these networks to perform design augmentation. For example, automatically adding functionality to existing object forms. This may prove extremely effective in the field of design automation, and exploring the design space of 3D objects. In particular, Variational Autoencoders and Adversarial Networks have been employed for this process with success. Other future work may leverage textual analysis capabilities to help generate descriptions of what functions each object can perform. Both of these will take advantage of the extremely high parameter count of neural networks which could assist in using a model to generate new objects or describe existing ones. Another active area of research is interpreting neural networks. As discussed, though there are some methods commonly used for 2-D input, the field is much less mature for the 3-D domain. This would be useful for more complex objects, which would benefit from demonstrating partial understanding of an object.

Other future work could compare the ability of 2-dimensional image data of these rendered objects on this task, which would enable the use of larger resolution inputs. Input may also be augmented with metadata such as scale, and even a per-voxel annotation of material to achieve a more complete prediction. Future work could also consider more complex functionalities, or modeling interactions between objects. Finally, future work should explore alternative methods of representing these objects, as the voxelization process is both time consuming and imprecise. While some work has proposed using different data structures for sparse voxelized objects, another approach may be to analyze object mesh directly, as a sequence of surfaces, since continuity can easily be lost.

7 Acknowledgment

This research is funded in part by the National Science Foundation grants: NSF NRI 1527148 and NSF DUE 1449650. Any opinions, findings, or conclusions found in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] Vasudevan, N., and Tucker, C. S., 2013. "Digital representation of physical artifacts: The effect of low cost, high accuracy 3d scanning technologies on engineering education, student learning and design evaluation". In ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V001T04A013–V001T04A013.

[2] Tucker, C. S., Saint John, D. B., Behoora, I., and Marcireau, A., 2014. "Open source 3d scanning and printing for design capture and realization". In ASME 2014 International Design Engineering Technical Conferences and Com-

puters and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V01BT02A013–V01BT02A013.
[3] Shyy, W., Udaykumar, H. S., Rao, M. M., and Smith, R. W., 2012. *Computational fluid dynamics with moving boundaries*. Courier Corporation.
[4] Buede, D. M., 2011. *The engineering design of systems: Models and methods*. John Wiley & Sons.
[5] Gero, J. S., and Kannengiesser, U., 2004. "The situated function–behaviour–structure framework". *Design studies,* **25**(4), pp. 373–391.
[6] Chandrasegaran, S. K., Ramani, K., Sriram, R. D., Horváth, I., Bernard, A., Harik, R. F., and Gao, W., 2013. "The evolution, challenges, and future of knowledge representation in product design systems". *Computer-aided design,* **45**(2), pp. 204–228.
[7] Chen, Y., Liu, Z.-L., and Xie, Y.-B., 2012. "A knowledge-based framework for creative conceptual design of multi-disciplinary systems". *Computer-Aided Design,* **44**(2), pp. 146–153.
[8] Qi, J., Hu, J., Zhu, G., and Peng, Y., 2015. "Automatically synthesizing principle solutions in multi-disciplinary conceptual design with functional and structural knowledge". In ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V02AT03A006–V02AT03A006.
[9] Bhatt, M., Hois, J., and Kutz, O., 2012. "Ontological modelling of form and function for architectural design". *Applied Ontology,* **7**(3), pp. 233–267.
[10] Townsend, J. D., Kang, W., Montoya, M. M., and Calantone, R. J., 2013. "Brand-specific design effects: form and function". *Journal of Product Innovation Management,* **30**(5), pp. 994–1008.
[11] Kang, S. W., and Tucker, C. S., 2015. "Automated concept generation based on function-form synthesis". In ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V02AT03A008–V02AT03A008.
[12] Tseng, I., Cagan, J., Kotovsky, K., and Wood, M., 2013. "Form function fidelity". *Journal of Mechanical Design,* **135**(1), p. 011006.
[13] Roy, U., Pramanik, N., Sudarsan, R., Sriram, R. D., and Lyons, K. W., 2001. "Function-to-form mapping: model, representation and applications in design synthesis". *Computer-Aided Design,* **33**(10), pp. 699–719.
[14] Tsai, H.-C., Hsiao, S.-W., and Hung, F.-K., 2006. "An image evaluation approach for parameter-based product form and color design". *Computer-Aided Design,* **38**(2), pp. 157–171.
[15] Sylcott, B., Cagan, J., and Tabibnia, G., 2013. "Understanding consumer tradeoffs between form and function through metaconjoint and cognitive neuroscience analyses". *Journal of Mechanical Design,* **135**(10), p. 101002.
[16] Sylcott, B., and Cagan, J., 2014. "Modeling aggregate choice for form and function through metaconjoint analysis". *Journal of Mechanical Design,* **136**(12), p. 124501.
[17] Orsborn, S., Cagan, J., and Boatwright, P., 2009. "Quantifying aesthetic form preference in a utility function". *Journal of Mechanical Design,* **131**(6), p. 061001.
[18] Tseng, I., Cagan, J., and Kotovsky, K., 2012. "Concurrent optimization of computationally learned stylistic form and functional goals". *Journal of Mechanical Design,* **134**(11), p. 111006.
[19] Cheong, H., Chiu, I., Shu, L., Stone, R., and McAdams, D., 2011. "Biologically meaningful keywords for functional terms of the functional basis". *Journal of Mechanical Design,* **133**(2), p. 021007.
[20] Lowe, D. G., 1999. "Object recognition from local scale-invariant features". In Computer vision, 1999. The proceedings of the seventh IEEE international conference on, Vol. 2, Ieee, pp. 1150–1157.
[21] Bay, H., Tuytelaars, T., and Van Gool, L., 2006. "Surf: Speeded up robust features". *Computer vision–ECCV 2006*, pp. 404–417.
[22] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G., 2011. "Orb: An efficient alternative to sift or surf". In Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, pp. 2564–2571.
[23] Dalal, N., and Triggs, B., 2005. "Histograms of oriented gradients for human detection". In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1, IEEE, pp. 886–893.
[24] Tombari, F., Salti, S., and Di Stefano, L., 2011. "A combined texture-shape descriptor for enhanced 3d feature matching". In Image Processing (ICIP), 2011 18th IEEE International Conference on, IEEE, pp. 809–812.
[25] Song, M., Sun, Z., Liu, K., and Lang, X., 2015. "Iterative 3d shape classification by online metric learning". *Computer Aided Geometric Design,* **35**, pp. 192–205.
[26] Cortes, C., and Vapnik, V., 1995. "Support-vector networks". *Machine learning,* **20**(3), pp. 273–297.
[27] Breiman, L., 2001. "Random forests". *Machine learning,* **45**(1), pp. 5–32.
[28] 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**(3), pp. 211–252.
[29] Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012. "Imagenet classification with deep convolutional neural networks". In Advances in neural information processing systems, pp. 1097–1105.

[30] Simonyan, K., and Zisserman, A., 2014. "Very deep convolutional networks for large-scale image recognition". *CoRR,* **abs/1409.1556**.
[31] He, K., Zhang, X., Ren, S., and Sun, J., 2016. "Deep residual learning for image recognition". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778.
[32] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J., 2015. "3d shapenets: A deep representation for volumetric shapes". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1912–1920.
[33] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., et al., 2015. ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR].
[34] Maturana, D., and Scherer, S., 2015. "Voxnet: A 3d convolutional neural network for real-time object recognition". In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE, pp. 922–928.
[35] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E., 2015. "Multi-view convolutional neural networks for 3d shape recognition". In Proceedings of the IEEE international conference on computer vision, pp. 945–953.
[36] Shi, B., Bai, S., Zhou, Z., and Bai, X., 2015. "Deeppano: Deep panoramic representation for 3-d shape recognition". *IEEE Signal Processing Letters,* **22**(12), pp. 2339–2343.
[37] Su, H., Qi, C. R., Li, Y., and Guibas, L. J., 2015. "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views". In Proceedings of the IEEE International Conference on Computer Vision, pp. 2686–2694.
[38] Boscaini, D., Masci, J., Rodolà, E., and Bronstein, M., 2016. "Learning shape correspondence with anisotropic convolutional neural networks". In Advances in Neural Information Processing Systems, pp. 3189–3197.
[39] Shu, Z., Qi, C., Xin, S., Hu, C., Wang, L., Zhang, Y., and Liu, L., 2016. "Unsupervised 3d shape segmentation and co-segmentation via deep learning". *Computer Aided Geometric Design,* **43**, pp. 39–52.
[40] Zhang, Y., Bai, M., Kohli, P., Izadi, S., and Xiao, J., 2016. "Deepcontext: Context-encoding neural pathways for 3d holistic scene understanding". *arXiv preprint arXiv:1603.04922*.
[41] Maas, A. L., Hannun, A. Y., and Ng, A. Y., 2013. "Rectifier nonlinearities improve neural network acoustic models". In Proc. ICML, Vol. 30.
[42] Glorot, X., and Bengio, Y., 2010. "Understanding the difficulty of training deep feedforward neural networks.". In Aistats, Vol. 9, pp. 249–256.
[43] Rothe, R., Timofte, R., and Van Gool, L., 2015. "Dex: Deep expectation of apparent age from a single image". In Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 10–15.
[44] Nooruddin, F. S., and Turk, G., 2003. "Simplification and repair of polygonal models using volumetric techniques". *IEEE Transactions on Visualization and Computer Graphics,* **9**(2), pp. 191–205.
[45] Riegler, G., Ulusoy, A. O., and Geiger, A., 2017. "Octnet: Learning deep 3d representations at high resolutions". In CVPR.
[46] Chollet, F., 2014. Keras: Deep Learning library for Theano and TensorFlow. Tech. rep.
[47] Theano Development Team, 2016. "Theano: A Python framework for fast computation of mathematical expressions". *arXiv e-prints,* **abs/1605.02688**, May.
[48] Kingma, D., and Ba, J., 2014. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*.
[49] Zeiler, M. D., and Fergus, R., 2014. "Visualizing and understanding convolutional networks". In European conference on computer vision, Springer, pp. 818–833.

List of Tables

List of Figures