Classifying 3-Dimensional Shapes with Convolutional Neural Networks (CS 760 2017)

David Porfirio [1] **Andrew Schoen** [1] **Lorenzo Najt** [1] **Krittisak Chaiyakul** [1] **Junda Sheng** [1]

Abstract

We investigate various parameters for the structure and implementation of Convolutional Neural Nets (CNNs), offering insights into the capabilities and limitations of such algorithms in the space of 3-dimensional shape recognition. In particular, we train and test 3-dimensional spheres and tori that have undergone a range of standard distortions. Our results indicate that such a task is clearly possible for some collections of individual distortions, but further work is needed to improve classification accuracy on composites of distortions, and to elucidate the precise features learned by such systems in this domain.

1. Introduction and Motivation

Humans have evolved a complex classification scheme to recognize everyday, complex 3-dimensional objects. These classifications are usually independent of things such as scaling, location, or texture. Sometimes, they are also independent of stretching or warping. For example, the "ball" shape includes tennis balls, volleyballs, footballs and the moon, despite the fact that they differ in texture, warping, and scale. A great deal of research in psychology, especially cognitive psychology, has been done to establish how humans perform these tasks on a daily basis with exemplar/non-exemplar learning (Tennyson et al., 1972; Sinha et al., 2006).

Due to the human ability to detect and discriminate between classes of shapes, we seek to assess the ability of Convolutional Neural Networks (CNNs) to learn this classification problem. In doing so, our work seeks to establish optimal parameters when designing such systems, as well as evaluate their limitations and capabilities in this domain.

Our work culminates in three contributions to the computational classification of shapes. First, we conclude that the classification of spheres and tori is possible with CNNs, and that the different shapes are easily distinguishable when individual distortions are applied to the shapes. However, when distortions are combined, the CNN has difficulties classifying the shapes. This, of course, depends on the architecture of the net and how the shapes are represented in 3-dimensional space. Secondly, we present ways to represent these shapes in 3-dimensional space, and evaluate their efficacy. And finally, although this was not the primary focus of our work, we present a CNN architecture and the parameters that we used for 3-dimensional shape classification.

The motivations for this work include a general interest in image-based classification and shape detection. It also has potential applications to the medical imaging fields, where learning to classify complex shapes such as brains of healthy or non-healthy individuals which differ with regards to a set of deformations could be useful for diagnostics. Additionally, there are clear applications to the field of facial detection, where similar constraints occur.

2. Related Work

2.1. Human Studies

This research is founded on the ability of humans to perform discrimination of different 3-dimensional shapes, despite distortions of those shapes in ways such as translation (where it is located in our vision or the world), rotation (the angle of viewing), linear warping (balls to footballs), polynomial warping (spheres to paraboloids) or other transformations (balls to cubes). We know from previous research on humans that it is possible to discriminate shapes such as faces with very low resolution, and a holistic method is used to some extent, but this develops over time from rudimentary elementary features (Sinha et al., 2006).

This is potentially informative to how a CNN may learn the distinction between shapes, as the convolutional layers help to capture features in a non-location-specific manner to be assembled in higher levels.

2.2. Convolutional Neural Nets

CNNs are a type of Neural Networks that specialize in classification and recognition of inputs which contain repeated features, but not necessarily a strict organization of those features. Their models are based on animal visual percep-

tion, which builds a visual image up from basic lines and angles. These networks have been used in handwriting and letter recognition, (LeCun et al., 1998). These networks are now being used regularly in image classification tasks (Krizhevsky et al., 2012). New innovations in tiled convolutional layers improve robustness for invariances by building it into the structure of the net by tying together weights (Le et al., 2010).

2.3. Topology and neural codes

Laboratory experiments have shown that mice brains do topological computations in order to understand the shape of their environment, (Curto, 2016). We see this as evidence that ideas from topology may be important for understanding neural networks.

2.4. Mathematical Preliminaries

Before moving on to the topic of this study, it will be useful to provide a small amount of mathematical context to the shape discussion.

For this, $\mathbb{R}^3$ denotes 3-dimensional Euclidean space, with coordinates x,y and z.

2.4.1. Sphere

The standard sphere is the set of (x, y, z) such that $x^2 + y^2 + z^2 = 1$. See Figure 1.

The standard 3D Gaussian is the random vector in $\mathbb{R}^3$ with x, y, z components each given by an independent Gaussian $N(0, 1)$.

To sample uniformly from a sphere first sample a standard 3D Gaussian, then normalize: $x \sim N(0, I)$, and $\frac{x}{||x||}$. This gives a probability distribution on the surface of the sphere. Since the probability density for standard Gaussian is invariant under rotations, the resulting distribution on the sphere is invariant under rotations - this is the sense in which it is uniform.

2.4.2. Torus

The standard torus $T_{r,R}$ is parametrized by $\psi(\theta, \phi) = ((R + rcos(\theta))cos(\phi), (R + rcos(\theta))sin(\phi), rsin(\theta))$ for $\theta, \phi \in [0, 2\pi]$, for some r and R with $R > r > 0$. Figure 1.

To sample uniformly from the torus, draw θ and ϕ according to the following distribution: uniform on ϕ in $[0, 2\pi)$ and independently θ according to the density $\frac{1}{2\pi}(1 + \frac{r}{R}cos(\theta))$ for $0 \leq \theta < 2\pi$.

This distribution compensates for the fact that the inner band of the torus has less area than the outer band. It is computed using methods from calculus. For more detail, see (Diaconis et al., 2012).

2.4.3. Simplicial shapes

For one experiment, we also studied cubes, a tetrahedron, and a "square torus" (a cube with a square tunnel through it), Figure 1. Simplicial homology is a method from algberaic topology that associates to a surface a sequence of vector spaces, with dimensions counting the number of holes of a certain dimensional-size. See any introductory text on algebraic topology for a thorough explanation of this. From the "simplicial homology" point of view, cubes and tetrahedrons should be labelled as spheres since they have one "2-dimensional hole" and 0 "1-dimensional" holes. From the same point of view, the square torus should be a torus. But from other points of view, tetrahedrons should be labelled as cubes - perhaps because they have sharp corners.

3. Methods

For this study, datasets were generated for each test based on the procedure in this section. Unless otherwise noted, training and testing sets were drawn from the same populations.

3.1. Shape Generation

In our study we will consider only surfaces embedded inside a cube. For each of these surfaces, there is a method for sampling from its points uniformly randomly - see mathematical preliminaries above.

Since we are doing computations, when we speak of a surface we will mean implicitly a cloud of points drawn from that surface in a uniform manner.

3.2. Creating pointcloud data

Let S be the standard sphere, and $T = T_{r,R}$ the standard torus.

We made N draws of M points from S and T independently and uniformly - S_n denotes the nth draw of M points from S.

Throughout we will call M the sampling rate. We will try $M = 100, 1000, 10000$ in our computations.

3.3. Distortions

Since humans can identify shapes after they have been distorted or moved around, we want to see if our neural net will learn to classify the shapes even after these distorations have been applied. In order to generate data for experiments, we create random distortions systematically using group theory. These distortions, drawn randomly accord-

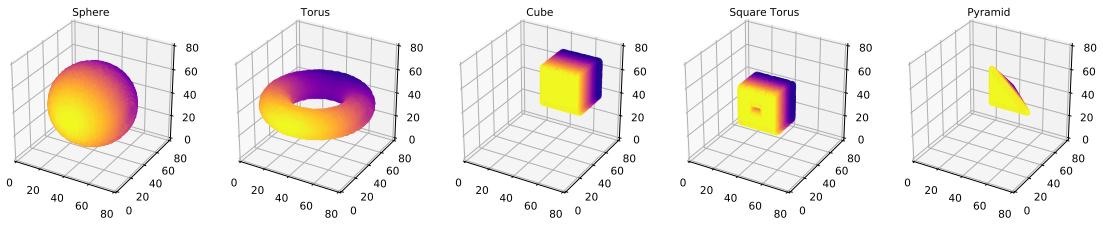

Figure 1. A visualization of the various shapes generated. Unless otherwise noted, only spheres and tori were used as a binary decision-making task.

ing to a procedure outlined below, are applied individually to each training instance as functions on the point clouds generated above. A visualization of these can be found in Figures 2 and 3.

(To be precise: An individual experiment works with a fixed collection of distortions F, and there is a notion of a random $f \in F$. We sampled A_n and B_n from F independently and randomly. By $A_n(S_n)$ we mean A_n applied to the sampled pointcloud S_n, likewise for B_nT_n. Our pointcloud dataset was $(A_n(S_n), sphere)$ and $(B_n(T_n), torus)$, where $sphere$ and $torus$ were the class labels.)

We used the following kinds of distortions as parameters in our data creation. The input values to generate each distortion are shown in Figure 4

3.3.1. Default

A rather uninteresting distortion, this is essentially equivalent to the identity matrix, leaving the original shape intact. While not particularly interesting itself, it provides a baseline for the other distortions.

3.3.2. Scaling

We drew a random scalar in $N(0, \sigma^2)$ to scale our shapes.

3.3.3. Rotations

$SO(3)$, the "special orthogonal group," is the collection of rotation matrices in $\mathbb{R}^3$. Since the composition of two rotations is a rotation, and since the inverse of a rotation is a rotation, this collection forms a group - an algebraic structure capturing the idea of symmetry.

Group theory assures us that there is a notion of drawing randomly, uniformly from $SO(3)$. It can be concretely realized by drawing 3 vectors independently from the standard 3D gaussian, and performing the Grahm-Schmidt process.

3.3.4. Translations

To translate our shapes, we drew a random translation vector from the standard $3D$ Gaussian, with entries drawn from $N(0, \sigma^2)$.

3.3.5. Linear Warps

In implementation, we also created the option to draw $A \in SO(3)$ according to the above proceedure and also add fuzziness to it.This meant that we created a $3x3$ matrix, with entries drawn from a $N(0, \sigma^2)$, with small σ^2.

3.3.6. Polynomial Warps

Let $p(x, y) = x^2 + y^2$, and let $a > 0$. Define $g_a(x, y, z) = (x, y, z + p(x/a, y/a))$, and $g_a^{-1}(x, y, z) = (x, y, z - p(x/a, y/a))$. (The polynomial p is not special, it is just the one that we did experiments with.)

We created another family of distortions by drawing a rotation matrix A randomly as in the previous subsection, and then applying $g \circ A \circ g^{-1}$ to our shapes.

3.3.7. Noise

For each point x in the surface, we made it noisy by adding to it a random $N(0, \sigma^2)$, with σ^2 small. The noise was drawn independently for each point.

3.3.8. All-Except-Polynomial

This is a compound distortion consisting of scaling, rotation, translation, linear warp, and noise. Poly was excluded in this set since there is some *a priori* rationale for believing that the addition of polynomial warps may render it untractable, but without remain tractable. Since we were interested if this were the case, this compound distortion was included, in addition to *all*.

3.3.9. All

This is a compound distortion consisting of scaling, rotation, translation, linear warp, polynomial warp, and noise.

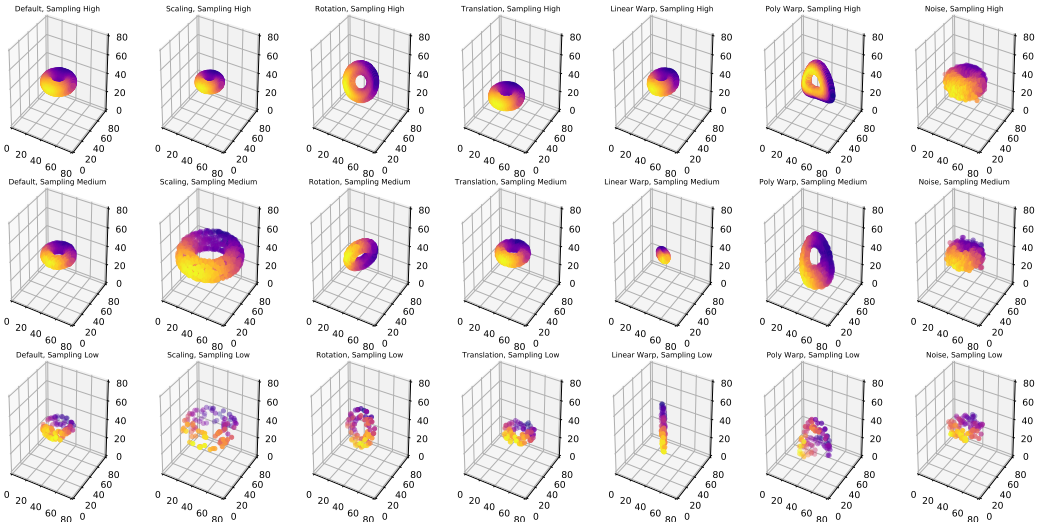

Figure 2. Examples of toruses with the various distortions applied individually, along with the different sampling frequencies.

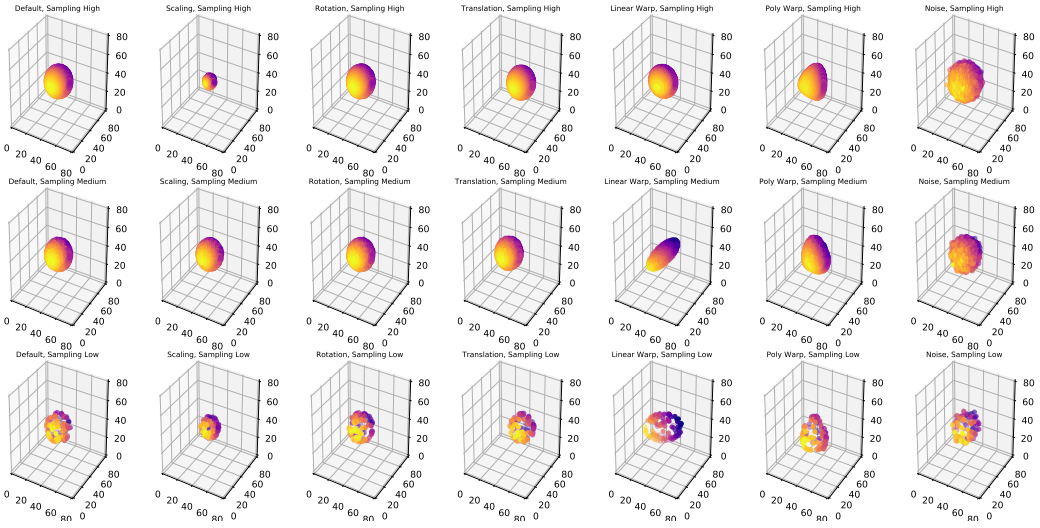

Figure 3. Examples of spheres with the various distortions applied individually, along with the different sampling frequencies.

Distortion	2-class Input	Multi-Class Input
Default	none	none
Scaling	$\sigma = 5$	$\sigma = 3$
Rotations	yes/no	yes
Translations	$\sigma = 4$	$\sigma = 4$
Linear Warps	$\sigma = 0.3$	$\sigma = 0.2$
Polynomial Warps	$\sigma = .5$	$\sigma = 0.3$
Noise	$\sigma = 1$	$\sigma = 1$

Figure 4. Input values for each of the nine distortions applied to the shapes.

It represents the most complex set of distortions we tested.

3.3.10. Settings for Each Distortion

3.4. Preparing data for input to the CNN

Data used to train the CNN was generated as point clouds in R^3 according to the aforementioned methods. Due to practical constraints, we keep our points on a grid, and limit our region to the unit cube I^3. (There may be fundamentally different approaches... see future directions below for some speculation.)

In particular, we divide our cube up into a grid of small cubes, or "voxels." Each voxel is a feature in the feature vector that serves as input to the CNN. If a sampled point on the surface of a shape lies in a particular voxel, then that voxel's input value is non-negative.

We investigated two methods for assigning non-negative input values for features that correspond to voxels containing one or more points on the surface of a shape. The first method, our default, is a *binary approach* that assigns a 1 to the features with no points in the corresponding voxel, and a 0 otherwise. This approach effectively answers the question, "does this voxel contain a point?" In contrast, the second method permits features to retain information about how many points their corresponding voxels contain. In this method, which we term the *float-based approach*, the input value assigned to feature i is simply the number of points in voxel i divided by the maximum number of points found in any given voxel. In other words, for the i^{th} voxel in the set of voxels V, $input_i = \frac{numPoints_i}{max(numPoints_j)\forall j \in V}$.

We hypothesize that the *float-based approach* will lead to higher testing accuracies in the CNN when noise is present, and sampling is high. This is due to its ability to encode the extent to which a shape passes through a voxel. For instance, in Figure 5 the feature corresponding to the Voxel 1 would have a higher input value than the feature corresponding to Voxel 2.

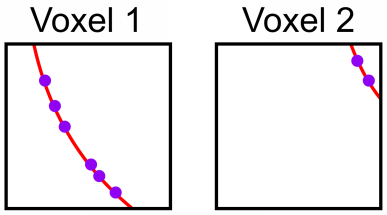

Figure 5. In this 2D example, the surface of a shape passes more through Voxel 1 than through Voxel 2, resulting in 6 points sampled for Voxel 1 and 2 points sampled for Voxel 2. Using the *float-based approach*, the feature corresponding to Voxel 1 would have 3× the input value as Voxel 2. Using the *binary* approach, both would have input values of 1.

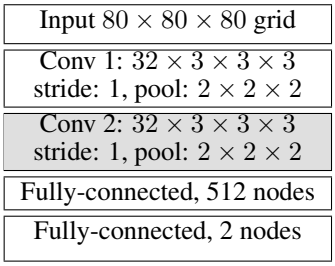

Figure 6. Structure of the CNN. Note that the second convolutional layer, Conv 2, is optional and thus grayed-out.

3.5. Learning Algorithm

We trained and tested our data on a 3-dimensional convolutional neural net built using the Keras deep learning library (Chollet et al., 2015). The parameters that we used for the CNN are in Table 1, and the structure of the CNN is shown in Figure 6. We varied the number of convolutional layers between 1 and 2, each with rectified linear unit (ReLU) activation functions, and max-pooled each convolutional layer with a $2 \times 2 \times 2$ pool size. We also added a 20% dropout rate for the convolutional layers. The number of learnable filters was set to 32 for each convolutional layer, with each filter having a $3 \times 3 \times 3$ filter size applied with a stride of $1 \times 1 \times 1$. Zero-padding was applied to each convolutional layer such that the output feature map size was equal to the size of the input. We also added two fully-connected layers after the convolutional layers, the first with 512 nodes, and the second with the number of nodes equal to the number of classes in our dataset. This value is always 2 for our datasets, but can be more in the event that additional shape-types are classified. For our backpropogation parameters, we used a learning rate of 0.01 and a momentum of 0.9. Finally, we trained our CNN with a batch-size of 32.

Finally, we employed an early stopping technique while training our data that resembled the early stopping technique of Lutz Prechelt, which utilizes the error of the val-

PARAMETER	VALUE
NUMBER OF CONV LAYERS	1 OR 2
CONV LAYER ACTIVATION	RELU
CONV LAYER DROPOUT	0.2
NUMBER OF FILTERS	32
FILTER SIZE	$3 \times 3 \times 3$
FILTER STRIDE	$1 \times 1 \times 1$
ZERO-PADDING	YES
POOLING TYPE	MAX POOL
POOL SIZE	$. 2 \times 2 \times 2$
LEARNING RATE	0.01
MOMENTUM	0.9
BATCH SIZE	32

Table 1. Summary of CNN parameters

idation set to determine when to stop (2012). Specifically, we stopped when the validation error showed no improvement after 3 epochs. Our primary reason for using early stopping was to take advantage of when the error of the testing set was likely to be the lowest under the assumption that the testing and validation set errors would be similar. Other reasons included to prevent overfitting of the training data, and to decrease the amount of time taken to train.

3.6. Runtime Environment

We were unable to run our CNN using UW-Madison's Center for High Throughput Computing. Therefore, training and testing the neural net took place on two personal computers – one laptop computer running Macintosh OSX with 16 GB of memory and a 2.3 GHz quad core processor, and one desktop computer running Microsoft Windows 7 with 16 GB of memory and a 3.6 GHz quad core processor. Each machine trained and tested separate CNNs, without sharing computing power.

3.7. Tests

3.7.1. MAIN ANALYSES

The main analyses consisted of a 9x3x2 grid, with levels of the aforementioned distortions, sampling rates of 100, 1000, and 10000, and neural nets with one or two convolutional layers. Accuracies and number of epochs before early stopping were measured for each analysis.

In order to train the net, we split our data into training and testing sets, with 480 instances initially in the training set and 120 instances in the testing set. The training set was further split into training and validation sets, with the final training set having 320 instances and the validation set having 160 instances. We did not perform k-fold cross validation due to limitations in computing power available to us. We did not increase the size of the training set for similar reasons.

3.7.2. MULTICLASS ANALYSIS

As a pilot, we also ran an analysis using all 5 shapes and 5-fold cross-validation, and an *all* composition with slightly eased parameters to see if the pattern of errors may indicate tactics of learning by the Neural Net. Like the main analyses, we split our data into training and testing sets, with 480 instances initially in the training set and 120 instances in the testing set. The training set was further split into training and validation sets, with the final training set having 320 instances and the validation set having 160 instances. To gauge the importance of training size in this domain, we also ran another test with double the training size, validation size, and testing size. Unlike the main analyses, all tests were done with the floating-point method of sampling, with a sampling rate of 10000.

4. Results

In order to discriminate the difference between input shape variation parameters and the capabilities of a CNN to learn them, we have separated our analyses into two major categories: Parameters, and Limitations and Capabilities.

Parameters refers to the different types of variations we can impose on the input data, such the types of shapes drawn from and the distortions which distinguish different instances, such as translation, rotation, etc. (see section 3.3). We also investigate the effect of different sampling rates on the shape surfaces on a logarithmic scaling from sampling 100 points (low) to 1000 points (medium), to 10000 points (high). All of the aforementioned parameters are varied with one and two convolutional layers using the *binary approach* for assigning input values to features corresponding to voxels. Finally, we performed four additional experiments which used the *float-based approach* of assigning input values to features corresponding to specific voxels. The experiments were performed with *translation*, *linear warp*, *all (except poly)*, and *all* parameter settings with a sampling rate of 10000 and one convolutional layer.

Regarding limitations and capabilities, we investigate the effect of the different parameters and input data on the CNN.

4.1. Parameters

Our results for each parameter setting are shown in Tables 2 and 3. Each table is a 9×3 parameter grid of tests for either one or two convolutional layers. These results contain both the % accuracy for each dataset and number of epochs taken to converge in these settings. The percent-accuracy results are summarized in Figure 7, which show the behav-

Parameter Tests	Sampling 100 (Low)		Sampling 1000 (Medium)		Sampling 10000 (High)			
	Accuracy	Epochs	Accuracy	Epochs	Accuracy	Epochs	Accuracy	Epochs
Default	100%	23	100%	5	100%	5		
Scaling	87.5%	6	100%	25	100%	12		
Rotation	49.17%	5	100%	5	100%	5		
Translation	50%	10	93.33%	10	92.5%	6	97.5%	12
Linear Warp	46.67%	5	95.83%	8	95%	8	95%	8
Poly Warp	52.5%	10	100%	6	100%	5		
Noise	100%	25	100%	6	100%	5		
All (Except Poly)	54.17%	5	45%	7	43.33%	9	52.5%	5
All	48.33%	13	100%	5	58.33%	7	55%	5

Table 2. A description of the different parameters and results for a neural net with a single convolutional layer. Results in blue indicate a float-based approach.

Parameter Tests	Sampling 100 (Low)		Sampling 1000 (Medium)		Sampling 10000 (High)			
	Accuracy	Epochs	Accuracy	Epochs	Accuracy	Epochs	Accuracy	Epochs
Default	100%	25	100%	5	100%	5		
Scaling	50.8%	5	99.17%	15	48.33%	5		
Rotation	49.17%	5	100%	8	100%	6		
Translation	50.83%	6	90%	7	95.83%	15		
Linear Warp	52.5%	5	50%	7	97.5%	19		
Poly Warp	53.33%	5	100%	12	100%	25		
Noise	100%	23	100%	6	100%	5		
All (Except Poly)	55.83%	7	50.8%	6	52.5%	6		
All	55%	6	54.17%	5	51.67%	10		

Table 3. A description of the different parameters and results for a neural net with two convolutional layers. No tests were run with a float-based approach.

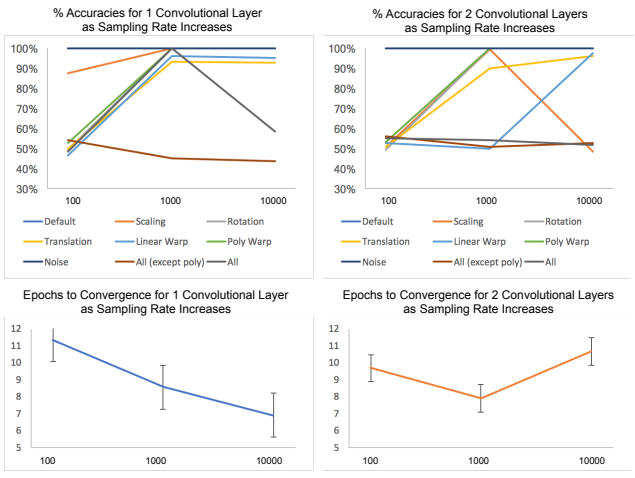

Figure 7. Testing accuracy plotted against sampling rate *(top)*, and time-to-convergence plotted against sampling rate *bottom*.

ior of the testing accuracies as the number of points sampled increases.

Figure 7 indicates that for CNNs with 1 convolutional layer, the testing accuracy increases for most parameter settings as the sampling rate increases. This finding does not hold when the *all* and *all (except poly)* distortions are applied. With *all*, the testing accuracy increases going from 100 samples to 1000 samples, but decreases with 10000 samples. This decrease may be due to the low number of instances used for training, and the fact that we did not train with k-fold cross validation. With *all (except poly)*, the testing accuracy when all dataset parameters are varied steadily decreases as the sampling rate increases. The accuracy results for 2 convolutional layers are more mixed. The accuracies for *all* and *all (except poly)* show little to no change at all. Additionally, the testing accuracy for *scaling* increases going from 100 samples to 1000 samples, but decreases with 10000 samples. Again, this decrease of accuracy may be a result of both the low number of instances used for training, and the fact that we did not apply k-fold cross validation.

Additionally, Figure 7 shows the effect on the number of epochs taken to converge as the sampling rate increases. The number of epochs taken to converge was calculated as the minimum of when early stopping occurred and 25 epochs. 25 epochs was the maximum number of epochs in which training could occur. On average, for both one and two convolutional layers, increasing the sampling rate decreases the time to convergence.

Lastly, Figure 7 shows that for the four experiments performed with the *float-based approach*, there was no significant difference in resulting testing accuracy. We thus performed no further trials with the *float-based approach*.

4.2. Capabilities and Limitations

We would expect the accuracy of the CNN on a testing set to increase as the sampling rate increases, since higher sampling rates lend to more accurate shape representations in voxel space. This is indeed the case in our results, minus a few instances where the accuracy was stagnant across sampling rates, and a few instances in which the accuracy increased briefly for 1000 samples, but decreased again for 10000. Suppose we attribute the latter to our small training sets and the fact that we did not employ k-fold cross validation in training the net. In other words, periodic poor performance of the net caused by our low training set size is not mitigated by higher performance of other folds. Then, it is indeed the case that higher sampling rates lead to more accurate classification of shapes. Therefore, the following discussion of the capabilities and limitations of the net will be limited to the experiments with a sampling rate of 10000 points.

From Table 2, it is clear that the CNNs which were trained and tested by varying zero or one shape distortion, such as *default*, *rotation*, or *scaling*, performed better than the *all* and *all (except poly)* parameter settings. This is also evident in the ROC curves for the experiments. Figure 8 shows two example ROC curves, one from the *translation* shape distortion and the other from the *all (except poly)* shape distortion sampled at 10000 points in a CNN with a single convolutional layer. These two results are chosen for analysis due to their being representative of distortions that the CNN learned well and not so well.

The ROC curve for *all (except poly)* clearly shows that the CNN is performing worse than chance. For most given false positive rates, the true positive rate is worse. Contrastingly, the ROC curve for *translation* shows a high true positive rate for most given false positive rates. The confusion matrices for both sets of parameters tells a similar story. For *all (except poly)*, the confusion matrix shows that each shape is more often being classified incorrectly than correctly, whereas the confusion matrix for *translation* shows shapes being overwhelmingly classified correctly.

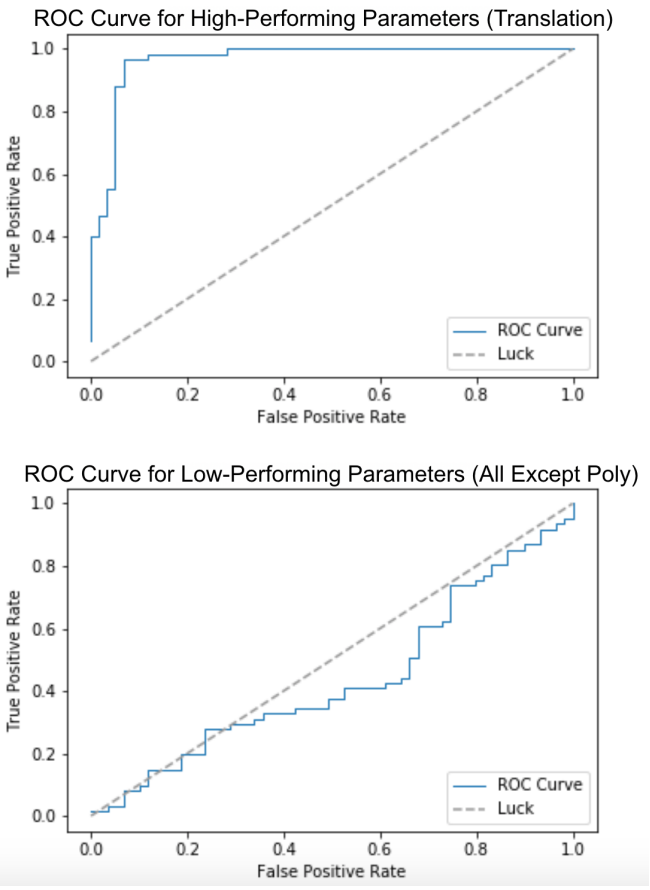

Figure 8. ROC curves for both a high and low-performing parameter settings.

	Torus True	Sphere True
Torus Prediction	28	27
Sphere Prediction	31	34

Figure 9. Confusion matrix for *all (except poly)* with a sampling rate of 10000 and a single convolutional layer

	Torus True	Sphere True
Torus Prediction	56	5
Sphere Prediction	4	55

Figure 10. Confusion matrix for *translation* with a sampling rate of 10000 and a single convolutional layer.

Additionally, the CNN overfit to the training set in the experiments with low testing set accuracies. In *all (except poly)*, for instance, the training set accuracy at early stopping was 97.81%, whereas the testing set accuracy was 43.33%. Similar discrepancies in training and testing sets are found for all other results with low testing set accuracies.

There is not much difference in all aspects of the results between the experiments with one or two convolutional layers. However, the experiments with two convolutional layers did take longer to converge than the experiments with one convolutional layer, in general (Figure 7). Similar to varying the number of convolutional layers, the few experiments that varied the *binary approac* and *float-based approach* to assigning non-negative input values to the CNN showed no significant difference in testing accuracies.

The above results indicate that in general, our method for training CNNs to classify 3-dimensional shapes is strong when classifying shapes that have undergone no distortions, or a single distortion so long as all shapes have undergone the same distortion in some way. However, our method falls short when classifying shapes that have undergone a composite of distortions, to the point that classification is no better than or worse than chance. Additionally, changing the number of convolutional layers from 1 to 2 results in no discernible difference in testing set accuracies.

4.3. Multiclass Analysis

Results from two 5-fold cross-validation runs on the multiclass discrimination task are shown in Table 4. In general, accuracies ranged from chance, 20% to 38%. Number of epochs before early stopping ranged from 5 to 7.

Training accuracy usually rose fast, despite the low test accuracy - sometimes as high as 80%. This coincided with an increasing validation error, causing the early stopping to engage. These two facts mean that overfitting was likely occurring.

As can be seen in Figure 11, an example confusion matrix from the *double-training-size* test is presented. From this matrix, it appears that the CNN is having difficulty distinguishing between smooth shapes (spheres and tori), and "edged" shapes (cubes, pyramids, and square tori) This specific example seems to indicate the presence of corners may be important.

5. Discussion

5.1. Efficacy

The neural net was mostly successful at learning the binary classification problems it faced when the distortions were non-composite. As mentioned before, increasing the sampling rate usually has the effect of improving accuracy. Interestingly, adding a second convolutional layer has a negligible if not detrimental effect on the neural net's efficacy.

The attempt of classifying between 5 different shapes was not as successful, achieving slightly better than chance. It is still impressive, given the algorithm was performing a quinternary discrimination task on a slightly easier version of parameters for a composite distortion that was effectively intractable for the binary discrimination task. Due to time constraints we were unable to study whether ensemble methods such as error-correcting codes (Dietterich & Bakiri, 1995) could improve efficacy for the binary or quinternary discrimination tasks, but this is a clear future avenue of testing.

5.2. Thoughts on PAC Learnability

Based on our computational results, we make the following (vague) conjecture:

Conjecture: Let G be a (compact) group of transformations of $\mathbb{R}^3$. Let S and T be surfaces. Then the problem of distinguishing gS and $g'T$ for uniformly random $g, g' \in G$ is (PAC) learnable by a (convolutional) neural network, provided there is no $g \in G$ so that gS and T are "very similar".

Compactness is a technical assumption that limits the "size" of the space of transformations. This includes $SO(3)$.

To be precise, our instance space is the space of n points $\mathbb{R}^3$, for n very large. This is called the configuration space of points in space, and is a very well studied object. For a surface S, the concept c_S is the set of n points that together lie on some from gS, for some $g \in G$, with notation as in the conjecture. The concept class C is the set of c_X, for $X \in \{S, T\}$.

In our experiment we drew samples by picking g randomly, and drew from gS uniformly, or from gT uniformly, and labelling as 1 or 0. General PAC learning would require drawing from an arbitrary distribution.

We claim that if $\epsilon, delta > 0$, there is a polynomial p in $1/\epsilon$ and $1/\delta$ (and possible other parameters, such as the dimension or complexity of G) so that we need p samples in order to produce, with probability $1 - \delta$ a neural net that has average error less than ϵ.

Making further progress on this conjecture would require further study of neural net outside the scope of this paper.

5.3. Mathematical Shape Classification - Isomorphism and invariants - What is our network learning?

Human mathematicians have developed two key concepts for classifying shapes. The first idea is a hierarchy of

PARAMETER TESTS	STANDARD TRAINING SIZE		DOUBLE TRAINING SIZE	
	ACCURACY	EPOCHS	ACCURACY	EPOCHS
FOLD 1	32.5%	6	33.33%	7
FOLD 2	21.67%	6	38.75%	6
FOLD 3	31.67%	5	32.5%	5
FOLD 4	29.17%	5	35%	6
FOLD 5	20.83%	5	20%	5

Table 4. Accuracy and epochs before early stopping for standard training size and double training size runs.

	Torus	Pyramid	Cube	Sphere	Square Torus
Torus Prediction	14	6	8	15	1
Pyramid Prediction	4	23	9	8	16
Cube Prediction	7	4	11	6	8
Sphere Prediction	15	2	1	14	3
Square Torus Prediction	7	17	19	6	16

Figure 11. A confusion matrix for the 5-class discrimination task.

thresholds for equivalences of shapes under some specified type of distortions; each level of the hierarchy is called a category. For example, two spheres A and B in 3-space with $radius(A) = 1$, $radius(B) = 2$ will be equivalent if we allow scaling and translations (working in "the category of spheres in 3-space with translations and scalings"), but not so if we only allow translations (working in "the category of spheres in 3-space with translations"). An equivalence is "witnessed" by a function of the allowed type, and this witness is called an isomorphism – in the example with spheres, the witness function would be the translation and scaling of 3-space that moves A onto B.

In general, given two shapes A and B, it may be valuable to prove or disprove that they are equivalent in some interesting category. Unfortunately, there are often many candidate distortions that could witness an equivalence of A and B, and it is impractical to check each one. This leads to the second key idea, which is the notion of invariant - in the examples above, we could see that sphere A is not equivalent to sphere B under $\{Translations\}$, because the radius doesn't change when we translate, and the radii of A and B are different. In this case, the radius of the sphere is an invariant of the equivalence relationship described by translations. Thus we say that the radius is a $\{Translations\}$-invariant, or is translation invariant.

In general, for any notion of equivalence, a primary strategy for showing that two shapes differ is to produce an invariant for that equivalence that the shapes differ on, especially when the space of candidate distortions between the shapes is too large to search through and exhaustively demonstrate the non-existence of a witness to equivalence.

Suppose we have two shapes A and B in 3-space, and some collection of distortions F, and suppose that there is some F-invariant of A and B, $I(A) \neq I(B)$. We feed a neural net the data: fA labeled with A and fB labeled with B for many random $f \in F$. If we hand our neural net some $f'A$ or $f'B$ and it correctly classifies it as A or B, we will say that the neural net has become F invariant. It is implausible that it checks every possible candidate witness, so we can wonder if it has has it learned to compute I, or some other invariant of F?

One way to gain some insight into this question is to change F to be a different set of distortions G, and see if the network is also G invariant. We can do this systematically by choosing a direction v to deform the transformation in, and see if the network remains $H_{t,v}$ invariant, for small t. (For example, choose a 1 parameter subgroup of $Diff(\mathbb{R}^3)$ $\phi(t)$, and use $H_{\phi,t} = \phi(t)\text{AffineGroup}(3)\phi(t)^{-1}$. We have to learn more about $Diff(\mathbb{R}^3)$.) We could also use this idea to produce a neural net which is more robust to changes in F, by picking many directions and building a training set using their transformations. We have started to study this question , but haven't had time to finish the computations.

Note that the idea of an invariants is for us really proba-

bilistic:Let I be some candidate computation from shapes to real numbers. Then it is possible that the distribution of $I(fA)$ for random $f \in F$ is far from the distribution of $I(fB)$ for random $f \in F$ (say in KL-divergence), but has some overlap. In this case, we may only be able to say that a mystery object is fA for some $f \in F$ with some probability, and $f'B$ for some $f' \in F$ with some other probability. (For example, if A is a big sphere, and B is a small sphere, then I could be average distance from the origin, and random F given by translation vectors drawn from a Gaussian distribution.) It is reasonable that our network has learned something along these lines, since it learns a probabilistic classifier.

We wonder if the invariants mathematicians have discovered to be useful are somehow implicit in the structure of a neural net, or if these sort of investigations could lead to useful new invariants for mathematicians to investigate. We didn't have time to computationally investigate this question sufficiently. However, our code is set up to do this, and we plan to continue to play with parameters.

5.4. Future Directions, Speculations and Open (to us) Questions

5.4.1. Parameters for CNNs

Due to compute time restrictions, we were unable to test a full array of parameters. Future work could look at increasing the dataset size, employing k-fold cross validation and tiling, and tweaking various CNN parameters. For the dataset size, we were restricted by the processing and memory requirements of our personal computers. We would therefore like to increase the number of instances used in the training and testing sets in conjunction with k-fold cross validation to increase the accuracy of the testing results. For pre-processing of the data, we would like to look into pre-processing the input data and tiling the CNN as a way to allow the net to learn different invariances (Le et al., 2010). Lastly, the approach investigated in this work focused mainly on varying parameters of the input data, rather than parameters of the CNN. Therefore, as a next step we would like to vary parameters of the net, starting with parameters that combat overfitting, such as dropout and pooling.

5.4.2. Storing shapes and moduli spaces

What is the right data type for teaching a computer about a shape? If we sample from a shape, what is the right way to store the pointcloud? Mathematically, we are satisfied with a set of points, but storing a set in a computer puts an ordering on the points, and the unnatural order contributes to noise. Our solution to this question was to store the shapes as a non-negative function on voxels.

We could phrase our algorithm as this diagram:
$\{\text{Pointcloud}\}$
$\downarrow_{\text{Estimator}}$
$M = \{\text{functions on voxels}\}$
$\downarrow_{\text{Neural Net}}$
$\{\text{Class of shape}\}$.

While we experimented with changing the estimator, it is plausible that we could experiment with M as well, and put in its place a space that is tailored to the geometric problem at hand. For example, if we were studying shapes cut out by quadratic algebraic equations in x, y, z, then we could investigate an maximum likelihood estimator onto the space of coefficients of quadratic equations in x, y, z - this is $\mathbb{R}^N$ with coordinates a_{ij} corresponding to the general quadratic equation $\Sigma_{i+j\leq 2} a_{ij} x^i y^j = 0$. The neural net would then take as input the coefficients of the quadratic equations, rather than the function on voxels. Such candidate spaces M are heavily studied in modern geometry, under the names parameter space and moduli space.

5.4.3. Symnets and Curvature invariants

If an ant wanders forever on a surface, then by keeping track of how the landscape curves around it, it will be able to compute whether it lives on a sphere or torus. This is a special case of the Gauss-Bonnet theorem. Mathematically, this is expressed as an integral over the surface of a measurement of the curving of the surface, called Gaussian curvature. The average amount of curving classifies a shape as either a sphere or a torus.

For any point on the unit sphere, the Gaussian curvature will be one – even though from the point of view of CNN, which allow only translational symmetries, the various regions of the sphere look different. However, if we allowed the feature window to rotate, it could recognize that each of these regions was fundamentally the same, and exploit that.

If we used a NN architecture that allowed for different transformations of the feature window, it's plausible that the machine could learn an approximation to Gaussian curvature on little patches of a surface, and average them to classify the shape as a torus or sphere.

Symnets (Gens & Domingos, 2014) are an approach to this problem that uses the entire affine group rather than just the translation group to build a neural net. We did not have time to explore this direction, but this is a potential avenue of future computational experimentation.

5.4.4. Persistent homology

Topological data analysis is a field of mathematics that concerns itself with building algorithms to detect the topology of pointclouds; their methods (in principle) accurately

solve the classification problem we posed here. We wonder about a comparison of highly trained neural net with their algorithms.

5.4.5. Generative Adversarial Networks

In the interest of further desciphering the feature learning of neural networks in the shape domain, a clear future direciton is the area of Generative Adversarial Networks (GANs) (Goodfellow, 2016). By employing a game-theory tactic of competing neural nets, one a generator and the other a descriminator, the result in image-based processing is a set of images representing "similar" concepts, such as letters or numbers (Kataoka et al., 2016). Using such a tactic, we could obtain examples of what these neural networks are considering equivalent, and thereby infer the underlying features they are relying on.

6. Conclusion

In the current study, a large array of parameters have been tested with regards to the implementation and efficacy of Convolutional Neural Nets (CNNs) in discriminating three-dimentional shapes such as spheres and tori. While there were constraints on computation which limited the numbers of trials and training size, we can conclude that certain distortions are more easily learned by a CNN. These include, as a general rule, individual distortions. When these distortions are combined, the ability of our CNN decreased rapidly to chance. As noted, this could be a product of small sample size, and may be remedied with a more robust training set.

Additionally, low sampling rate was usually detrimental to correct classification, likely because not enough data was present to get a clear enough rendering of the shape. Again, larger training sets may be helpful in this regard.

In conclusion, we present a first look at the capabilities and limitations of CNNs for the classification of 3-dimensional shapes, and offer suggestions on further research in this space.

References

Chollet, François et al. Keras. https://github.com/fchollet/keras, 2015.

Curto, C. What can topology tell us about the neural code? forthcoming. 2016. URL https://arxiv.org/abs/1605.01905.

Diaconis, Persi, Holmes, Susan, and Shashahani, Mehrad. Sampling from a manifold. *arXiv: 1206.6913*, 2012.

Dietterich, Thomas G and Bakiri, Ghulum. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1995.

Gens, Roberts and Domingos, Pedro. Deep symmetry networks. 2014.

Goodfellow, Ian. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Kataoka, Yuusuke, Matsubara, Takashi, and Uehara, Kuniaki. Image generation using generative adversarial networks and attention mechanism. In *Computer and Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference on*, pp. 1–6. IEEE, 2016. URL http://ieeexplore.ieee.org/document/7550880/.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutio pdf.

Le, Quoc V., Ngiam, Jiquan, Chen, Zhenghao, Chia, Daniel, Koh, Pang Wei, and Ng, Andrew Y. Tiled convolutional neural networks. In *In NIPS, in press*, 2010.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Prechelt, Lutz. *Early Stopping — But When?*, pp. 53–67. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35289-8. doi: 10.1007/978-3-642-35289-8_5. URL http://dx.doi.org/10.1007/978-3-642-35289-8_5.

Sinha, Pawan, Balas, Benjamin, Ostrovsky, Yuri, and Russell, Richard. Face recognition by humans: Nineteen results all computer vision researchers should know about. *Proceedings of the IEEE*, 94(11):1948–1962, 2006.

Tennyson, Robert D., Woolley, F. R., and Merrill, M. D. Exemplar and nonexampler variables which produce correct concept classification behavior and specified classification errors. *Journal of educational psychology*, 63(2):144–152, 04 1972. URL http://search.proquest.com.ezproxy.library.wisc.edu/docview/614320821?accountid=465.