Introduction

In recent years, researchers have looked to machine learning as a solution to otherwise challenging technical problems. Artificial neural networks (ANNs), the leading machine learning technique used today, build a function that transforms high-dimensional input data into a lower-dimensional output that (often, but not always) represents that input's classification. ANNs are a form of *supervised learning*, meaning that they "learn" by designing this function by examining existing input-output pairs and building the transformation based on these given training examples. One major challenge in setting up an ANN is in obtaining *labeled training data*: a dataset that labels existing input data with the desired classification. Traditionally, a large amount of human effort goes into creating such a dataset, since labeling collected input data often requires a human to classify (and/or validate the classification of) tens of thousands of data points.

In this project, we explore a new method for producing a training dataset based on *synthetic data generation*, called *Isolation-Based Scene Generation* (IBSG), a process that involves identifying and extracting key "ingredients" in the dataset and computationally compositing these ingredients to create artificial, automatically classified scenes. We compare neural networks trained using hand-labeled, real-world-collected data[1] with similar networks trained using both exclusively ISBG training data and a mixture of IBSG and real training data, and compare the accuracy of test results on the various ANN models that are trained. We explore when ISBG is an appropriate replacement for real data and make suggestions on how ISBG datasets should be created.

Background

Artificial Neural Networks

ANNs are loosely based on the neuronal structure of the brain, an idea which explains the "neural" in "neural network." In the brain, neuron cells are cells connected to each other through *synapses*: physical gaps between these cells across which electrical information is propagated. The typical neuron will have a synaptic connection to 100 to 100,000 other neurons.[2]

Synaptic connections between neurons transmit scalar information: the strength of the electrical impulse taking place between the cells. The receiving neuron is "programmed" to do different things with its input signals; this programming forms the backbone of the neuronal system. As an example, a neuron might fire only when it receives a certain amount of electrical input, e.g. if at least 500 of

[1]For the rest of this paper, data of this type will be described as "real data" or "real training data," for the sake of brevity.

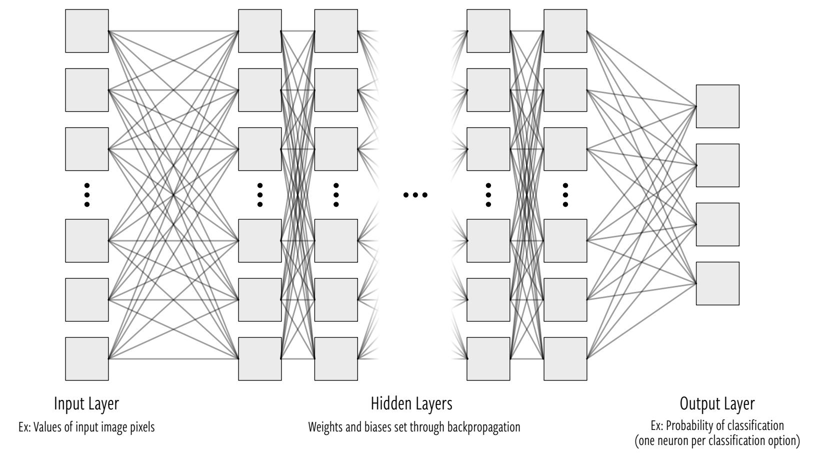
[2]Mehrota et. al., "Elements of Artificial Neural Networks," p. 8

its input neurons each transmit at least 80mV of electrical potential energy into that receiving neuron. With this example, we can see that:

1. Each neuron has many inputs and many outputs (called synapses)
2. The inputs of a single neuron can be represented as scalar values
3. Each neuron has a unique "activation function" that determines, for a given set of inputs, whether it will fire (send an impulse over its output synapses)

Artificial neural networks are computer programs written to follow similar principles as the neuronal system in the brain. Neurons and their connections are represented as nodes and edges in a graph. Each edge has a *weight*: a scalar value that determines the proportional strength between the output and the input. Each node has an *activation function*, a function that mathematically transforms the variety of input values into a single value, and a *bias*: a per-node modification of that value that determines how easily that node fires.

Feed-forward networks, one simpler type of ANN, are built from graphs whose nodes are connected in layers, where each node has an edge to every node in the preceding layer and every node in the following later (Fig. 1). The first "input" layer has node values set directly from the input information. The following nodes are organized in "hidden layers," which have little to no semantic meaning, but instead provide intermediate data transformation steps. Finally, the nodes in the "output" layer are set. These output neurons are usually defined in the training stage as corresponding to the different classifications that are trying to be learned; it is up to the implementer to decide how the training data will be labeled, and therefore, what output the ANN is testing for.


Input Layer
Ex: Values of input image pixels
Hidden Layers
Weights and biases set through backpropagation
Output Layer
Ex: Probability of classification
(one neuron per classification option)

Often, each node in a given layer will have the same activation function. These activation functions can have different purposes, such as condensing multiple values from certain prior nodes into a single value or performing a mathematical operation (often rounding or clamping) on a single value The configurations of the layers (i.e. the number of nodes in each layer and the activation function of each layer) are called *hyperparameters*: they are properties of the network itself and are set and fixed before the network sees any data.

The first feed-forward networks were hand-tuned: the researchers set the weights, biases, and activation functions by hand for every node and edge in the system.[3] In the 1970s and 80s, it was determined that the weights and biases of the system could be learned through *error back-propagation*,[4] a process known as *training* that involves:

1. randomly assigning biases for the neurons and weights for the edges,
2. testing the network on labeled training data,
3. calculating the error between the network's output and the data's annotations, and then
4. using gradient descent to modify the weights and activation functions so that the amount of error is reduced, and finally
5. repeating this process until the testing results start to show signs of *overfitting*, a phenomenon in which the training error rate and the testing error rate start to diverge, and the network becomes unable to perform well when given inputs from outside the training set.

After this training process is complete, the weights and biases of the neurons have been set in such a way to minimize potential testing error, and the ANN is ready to test new, unknown data points.

Artificial neural networks are at the heart of nearly all of today's machine learning implementations. They can take in a tremendous amount of seemingly unrelated input data, pick out relevant features (perhaps even features invisible to humans), and draw classification conclusions that are (ideally) similar to or better than those made by real people. They fuse together the reliability and speed of computers with the enigmatic classification ability of the human brain.

ANN Training Data

As described previously, one challenge in using ANNs is obtaining *labeled training data*: a dataset that both models the intended input data for the network and correlates each piece of data with the desired neural network output, were the network to receive that piece of data as input. ANN researchers maintain that a high-quality training data set will have three overarching characteristics:

1. There must be a large **quantity** of training data.

[3]TODO: Citation

[4]Error back-propagation was supposedly discovered independently by several researchers in the 1970s, and spread by Cun (1986), Parker (1985) and Rumelhart et. al. (1986) later. For more information on the history of neural networks, see Müller et. al (1995) and Fausett (1994).

2. The training data must be widely **varied**: it must represent each class with a diverse set of samples.
3. The dataset should exhibit **realism**: there should be very little difference between a representative sample of the training data set and a similar sample of the testing data.[5]

Picture of what bad diversity looks like, next to a picture of what bad realism looks like

Getting a data set with these qualities proves to be a challenging task; ensuring that this data set is properly labeled is even more difficult, since labeling each piece of data likely involves human supervision. However, the quality of the training data set directly impacts the success of the machine learning algorithm, so researchers will often go to great lengths to ensure that the variety and labeling of the training data set is accurate. Therefore, a traditionally well-implemented ANN has a large, varied, well-labeled training dataset backing it, a dataset that has had many man-hours put into maintaining its quality.

In recent history, researchers have been exploring the use of large, computer-generated data sets to train ML models. Instead of collecting and labeling real-world data, a computer algorithm generates a data set that mimics the data that might have been collected naturally. Since the computer algorithm generates each piece of data, it also has insight into specific features of the data, and can thereby label each piece of the data with the learned property as it is generated. With synthetic training data, the job of the supervisor is no longer to collect and label data; the computer will take care of both. Instead, the supervisor must ensure the training data generation algorithm outputs data with a wide enough variety to match the variety seen in testing, while also ensuring that the data remains realistic. For a synthetic training data set to be successful, it must have the same properties as a high-quality human-curated data set: namely, quantity, variety, and realism. The difficulty in creating such a model synthetically thereby falls to the programmer: they must encode the variety and unpredictability of the real world into a data-generating algorithm.

Synthetic training data is not a new idea. Several published applications of ML use synthetic data to either augment or replace the human-supervised data. But the method through which the synthetic data is generated can vary. Some studies use a statistics-based approach: finding the mean and standard deviation of numerical data and generating other data based on these values. Other studies that experiment on image data use 3D modeling to generate a scene and use special rendering techniques to make the render as realistic as possible. In this project, we explore a new data generation methodology using a system based on manipulating and compositing *isolations*: elements from a real-world scene that a human observes and extracts. We test the viability of using isolations to augment or replace hand-labeled data in different feed-forward ANN applications,

[5]These elements are discussed in further detail in §2.8 of Weiss & Kulikowsi's Computer Systems That Learn (1991), entitled "What Can Go Wrong?".

and explore the different ways isolation-based synthetic data can be generated.

The following section gives an overview of existing techniques and applications for synthetic data to train ANNs. Section 3 presents a formal definition of an isolation, indicates which form of data can be generated using isolations, and describes types of ANNs that work best with these data types. Section 4 sets up the experimental methodology for testing viability and presents its results. Section 5 proposes future work.

Prior Work

Prior research in the field of synthetic ANN training data has focused almost entirely on image analysis, in which a neural network is trained to detect certain features of 2D images. I present two common methods for synthetic data generation within the context of creating a vast quantity of labeled images to feed into a neural network: generation using 3D modeling and generation using 2D image composition.

3D Modeling

Shrivastava et. al. turned to synthetic data as a way of automatically gathering annotations for data: they used advanced 3D rendering techniques to generate realistic data while maintaining the annotations the model was able to attach.[6] This advanced rendering used Generative Adversarial Networks (GANs) as part of the rendering process to transform "perfect" 3D renders into degraded renders that more closely match the degradation seen in real-world data. They tested five scenarios, each combining different amounts of real data, traditionally-rendered synthetic data, and GAN-assisted rendered synthetic data. This study found that a model trained on traditionally-rendered synthetic data performed slightly worse than one trained on real data, but a model trained on GAN-assisted renders, especially when the amount of synthetic data was tripled, surpassed the model trained only on real data. In this case, with advanced rendering of 3D models, synthetic data can be a powerful addition to a training dataset.

Insert sample data from Shrivastava et. al.

Others still use 3D-modeled data from other sources with a similar result. In a 2016 paper, Richter et. al. used gameplay footage from *Grand Theft Auto*, *Watch Dogs*, and *Hitman* as a source of synthetic data and found that a model trained on 1/3 of the real data mixed with lots of gameplay footage performed better than a model trained on the complete real data training set.[7] This shows

[6]Shrivastava et. al., "Learning from Simulated and Unsupervised Images through Adversarial Training" (2016)

[7]Richter et. al., Playing for Data: Ground Truth from Computer Games (2016)

a large opportunity for synthetic data generation to replace much of the real-world data collection and labeling that takes place: in this way, large datasets can be generated faster and with greater accuracy. Clearly, there is some benefit to using synthetic training data: it can, in certain scenarios, outperform real data both in dataset collection time and in the resulting performance of the classifier.

Insert sample data from Richter et. al.

3D renders are a valuable source of synthetic training data in cases where real-world data cannot be collected. Hattori et. al. introduced a scenario in which a new surveillance system was installed and a model needed to be trained to detect pedestrians with no prior observations available.[8] The team rendered footage of a 3D environment that matched the viewpoint of the new camera, modeling the objects and buildings in the scene and introducing simulated pedestrians. They found that synthetic data models outperformed naïve pedestrian detection algorithms, models trained on hybrid data, and, in cases where real data is limited, even models trained on real data.

Insert sample data from Hattori et. al.

3D modeling can be a powerful tool for generating synthetic data; however, building a 3D model that mimics a scene can be labor intensive and requires specialized 3D modeling abilities. Furthermore, depending on the modeling and rendering capabilities of the 3D artist, the scene might not resemble testing data closely enough: note that the synthetic data described in the three studies above were either generated using advanced 3D modeling and rendering techniques (as seen in Shrivastava et. al. and Richter et. al.) or the fidelity of the 3D render was not important for training (as seen with Hattori et. al., who modeled a scene from a camera view far away). Where 3D rendering methods become less feasible, other data generation techniques have come into play.

Image Composition

Another method frequently used to synthetically generate image data is using 2D image composition. In this method, multiple 2D images are combined and mutated to create another 2D image, which becomes part of the data training set. Image composition is utilized by Jaderberg et. al., who studied synthetic data to train ANN models to recognize text in images.[9] To generate their training data, they used a six step process: 1) a word was rendered to an image using a given font; 2) that rendering was given a random border and shadow; 3) the background, border, shadow and text were colored using color clusters taken from real-world data; 4) the text, border, and shadow are randomly transformed using a perspective distortion; 5) the image is randomly blended with a crop

[8]Hattori et. al., "Learning Scene-Specific Pedestrian Detectors without Real Data." (2015)

[9]Jaderberg et. al., "Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition." (2014)

of a real image; 6) noise, blur, and compression artifacts are added. This data generation process trained a model that could outperform 3 other traditional word detection algorithms.

Insert sample data from Jaderberg et. al.

2D image composition can be especially powerful when combined with 3D modeling. Khungurn et. al. explored image composition while studying pose estimation of anime characters.[10] The team used pose information from 3D characters and rendered those characters to a 2D image. They then combined those character renders with artistic 2D backgrounds, using minimal transformation and composition techniques. They were able to build an estimator that could effectively estimate human poses in images that contained anime characters.

Insert sample data from Khungurn et. al.

Isolations and Synthetic Data Generation

I introduce a new method of synthetic data generation based on image composition and 3D modeling. This method, which I call Isolation-Based Scene Generation (IBSG), can be used to generate both visual and auditory detection models, and is optimized for minimal supervision while still providing quantity, variety, and realism. IBSG, I argue, works to bridge the gap between computational detection and human categorization, working with the best parts of both to create detection algorithms that are both effective and easy to build. To understand how IBSG is defined, however, it first must be explained how humans gather and process information about the world around them.

The Psychology of Detection

Humans are very good at using their senses—particularly sight and hearing—to infer things about the world and their current environment. Visual and auditory stimuli combined with intense, specific processing in the brain are the foundation for object recognition: the primary way we orient ourselves to a new environment is by trying to determine what we see and hear. It is these two types of data, sight and sound, that the brain has evolved to process incredibly well, since it is these types of data that give us the most information about our surroundings.

But what does it mean to "process" visual and auditory data? Psychologists recognize the importance of object recognition: the ability to extract features from stimuli and recognize an area of input as representing a certain object. In studying the brain, scientists have determined that humans search for specific

[10]Khungurn et. al., "Pose Estimation of Anime/Manga Characters: A Case for Synthetic Data" (2016)

features of stimuli: defining or distinct characteristics of a whole object. The brain then analyzes those features and tries to understand what known category that object belongs to. There are two leading theories behind categorization that elucidate how the brain thinks about a scene—exemplar theory and prototype theory—both of which have implications for how data is modeled in a neural network training dataset.[11]

Exemplar theory states that as we go about our lives, we build up a library of objects in our brain that are already categorized: when we see a new type of fish, we store it as a fish. When we see a new object, we compare it in parallel to every stored object before and make a comparison: if the object's features match up with the features in almost every fish we've ever seen, we can be confident it's a fish. Conversely, if a new object doesn't resemble any instance of a certain category, we have trouble categorizing it as such.

Prototype theory is similar, but argues for a center-out comparison model characterized by pre-computation. As we see different types of fish, the brain build up an idealized prototype fish that is the "average" of every fish it has seen. New observations are compared against these prototypes: the farther away the observation is from the prototype, the more difficulty we have detecting it.'

Prototype theory and exemplar theory are similar in practice, but exemplar theory argues for a system in which an object is categorized based on a category's boundaries: if we've seen another fish that resembles this fish, the object is within the *fish* category's bounds, and can be classified as such. Prototype theory instead builds a model around a category's center, and every new object is judged by how close or far it is from that center: in this way, an object is either more fish-like or less fish-like.

It happens to be the two types of data that humans are best at classifying—auditory and visual data—that computers have the most difficulty classifying. Ever since the earliest computers were built, data has been modeled as numbers in a one-dimensional list. This format is particularly bad for defining visual and auditory data, both of which rely on spatial dimensions: vision is a 2D projection of a 3D world, and sound is only given meaning when given a range of frequency and time. It proves difficult for computers, which store values in one dimension only, to computationally recognize features in two or more dimensional space.

Much of today's ANN research is aimed at giving computers the ability to classify auditory or visual data to a similar degree of accuracy with which humans can perform that same classification. I argue, in fact, that audio and image classification is an area of research that further bridges the gap between humans and computers.

It is these data types for which neural network based recognition can have the most impact; as seen in prior research, neural networks can effectively process

[11]Exemplar and Prototype theory knowledge gleaned from Reisberg, "Cognition," 6th ed.

auditory and visual data to recognize objects in the scene with a similar degree of accuracy to human recognition.

Isolation-Based Scene Generation

IBSG is a method of building artificial datasets of images or audio made for training ANNs. An IBSG dataset is created through a series of three steps.

First, the researcher either obtains some sample data or imagines what such data would look like. This small dataset will contain “scenes:” individual data points that take the form of a single image or a single audio clip. From these scenes, the researcher identifies *isolations*: individual components of these scenes that correspond to individual entities the brain can recognize as distinct. For most visual inputs, identifying isolations can be relatively simple: the physical objects in the image are all isolations; however, elements like lens flares would also count. For audio scenes, isolations can be a sounds from individual sources that can be reproduced on their own, like a ticking clock, a piece of music, a car driving by, or a sample of one’s voice. Importantly, though, the isolations chosen by the researcher should be present in different forms across the different scenes in the sample data.

Insert example of scenes with isolations highlighted

The researcher then (almost artistically) extracts those isolations or, if possible, records extracted samples. These extracted isolations might be transparent cutouts of objects, texture layers, or background layers for visual data. They might be individual recordings of noise, foley, dialog, or other sounds for audio data. An essential part of isolation generation is that the isolations themselves are parts of a scene that can be easily categorized by humans: it is this quality that allows IBSG to drastically simplify model generation, since the human can quickly define the isolation and its requisite modifications.

Insert example of isolations extracted

With these isolations identified and prepared, the researcher must create a computer program to composite these isolations randomly. This program should also introduce randomized modifications to the image. Isolations are placed, sized, and layered randomly. But they are also modified as individual inputs before and after composition: an image isolation might be blurred, darkened, lightened, sheared, projected, or otherwise transformed. An audio isolation might have its volume changed arbitrarily over time, might be pitch shifted, might be compressed or muffled or distorted. The final scene may be distorted in similar ways. The overall randomness of the composition is up to the researcher; one might strive for ultimate realism by introducing limitations on the modifications (like ensuring that humans are never bigger than trees); however, this is not necessary. Ultimately, the nature of the distortion, much like the nature of the composition, must be determined by the human making the artificial dataset; this decision will be much more artistic than it will be procedural.

Insert example of isolation modifications, global modifications, and how they can be combined

When IBSG is Useful

Part of the usefulness of an IBSG dataset is that the computer knows the location of each individual isolation within the output. While creating the dataset, the computer can generate not only each output scene, but also an annotation that uses this information. It becomes imperative that one isolation is the object that the ANN is trying to recognize: a ball-detection algorithm based on IBSG would have a ball as an isolation; a music-detection algorithm would have music as an isolation. IBSG datasets are useful training datasets, therefore, in situations where researchers are training an ANN to detect the location of a single isolation, especially if that isolation is covered, manipulated, or otherwise obscured.

Experimental Methodology

The experiment carried out in this project is meant to test the viability of IBSG synthetic data for training ANNs. I plan to create and test IBSG datasets against real-world datasets with different scene types, data types, and IBSG algorithms. More specifically, I will first take existing datasets built for training detectors and explore how the data could be modeled as an IBSG dataset. I will then train neural networks having the same hyper-parameters (layer configuration and size) with different ratios of real and IBSG training data and examine whether or not the IBSG data increases or decreases the accuracy of the ANN. I will use this data to draw conclusions about the usefulness of the IBSG method in training detectors.

Controlled Variables

Beyond the scope of this project is testing different neural network configurations based on data type. It could be reasonably hypothesized that one could tune network hyper-parameters for IBSG-specific training data; however, I limit the scope of the project to testing IBSG as a drop-in replacement for real training data. For each dataset, I will find an ANN configuration based on an architecture that works reasonably well (at my discretion), and will keep those hyper-parameters constant throughout that dataset's testing. I will use the same computer and training time for each model generation, and will use the same neural network framework (Tensorflow 1.12) throughout.

Independent Variables

There will be several dimensions through which I will test the viability of IBSG data.

I will generate IBSG training datasets based on four real-world datasets:

- Street View House Numbers (SVHN)
- Common Objects in Context (COCO)
- FreeField 1010 (Bird Sounds)
- OpenSLR (Dictated English speech)

In composing datasets, I aim to test two things: first, whether a quantity of IBSG data can reach the accuracy level of a similar quantity of real data; and, second, whether accuracy is improved when drastically more IBSG data is included. Therefore, I will generate training datasets that are composed of different real-to-IBSG ratios, some aiming to be the size of the original training dataset, some aiming to be larger.

In the list below, the first percentage is the amount of total real data sampled in the training dataset, and the second percentage is the amount of IBSG data that will be created compared to the total amount of real data available.

- 100% of real
- 80% of real, 20% IBSG
- 60% of real, 40% IBSG
- 40% of real, 60% IBSG
- 20% of real, 80% IBSG
- 0% of real, 100% IBSG
- 80% of real, 220% IBSG
- 60% of real, 240% IBSG
- 40% of real, 260% IBSG
- 20% of real, 280% IBSG
- 0% of real, 300% IBSG

The next set of independent variables are meant to elucidate some best practices for generating IBSG data. The first independent variable in this set is the number of isolations used to generate the IBSG data. Since the actual number of isolations is based on the original training data, I will split this up into a low, medium, and high number of primitives.

Like the number of primitives, the amount of data modification will be determined by the qualities of the original data set. I will also split this into a small amount of modification, a medium amount of modification, and a large amount of modification.

Testing Plan

With each training dataset created from a permutation of the above independent variables, I will train an ANN, the hyper-parameters of which are fixed on a per-dataset basis. After the NN is trained for 24 hours on a Microway Quadputer system that contains four Intel Xeon E5-4620v2 2.6 GHz Eight Core CPUs, 256 Gb of DDR3 1600 MHz ECC/Registered Memory, and an NVidia K20 GPU card, I will test it on a test dataset provided by the original data source. In testing, I will quantify how many test cases the trained ANN will correctly recognize. With this data, I will compare the number of each correct tests from each ANN to draw my conclusions about the performance of IBSG data.