DeepHoliday: Deep Learning Holiday Image Manipulation

Noah Walsh, Ben Valois, Rick Djeuhon, Jake Hamilton, and Derek Windahl

Abstract— **This paper presents to the best of ones knowledge the first deep dream image manipulator trained on a custom dataset of holiday images. This work is based on the DeepDream model, where images in the convolutional layers are manipulated and adjusted rather than the networks weights and biases. The Keras Inception V3 network with a final fully connected layer is trained to classify holiday images. Then a fully convolutional V3 network with the previous networks trained weights is used to modify and output non-holiday images inputted by the user. Future goals of this project include expanding the dataset, training on other deep convolutional networks, and using other deep learning image manipulation architectures, such as deep convolutional generative adversarial networks.**

I. INTRODUCTION

Image manipulation has been an active area of research and application for many years, and is another field that has applied deep learning. It involves modifying an image by using numerous methods and techniques in order to achieve the desired results. The purpose of this project is to use the activations in a trained deep convolutional neural network to manipulate an inputted image in order to produce an image one would find to be more festive. The project utilizes deep dreaming, a process in which an image is manipulated by gradient ascent, rather than the common gradient descent used to finetune the weights and biases in the neural network.

One of the main contributors to image manipulation through deep learning is Google. They have developed a method called Deep Dream [5] that would take an image and return a morphed version of it. Deep Dreaming [5] originated out a of study to visualize the uncanny effectiveness of deep convolutional neural networks in computer vision projects and other applications. After the notable Google AI post which introduced the topic [5], the trend became viral, seeping in the mainstream, and urging Google to release Deep Dream code for public application. Other research groups continued to focus on projects using Deep Dreams gradient ascent technique [5] to visualize layers, and produce many visceral results. The goal of this project is to use deep dream image manipulation as a tool to produce specific changes to make the image holiday-themed, rather than studying the specific network. This task is the first observed case of training the Inception V3 [15] neural network on a custom holiday dataset. That networks weights and biases are saved and uploaded to a fully convolutional Inception V3 network for image manipulation. This paper discusses the hidden layers of convolutional neural networks in order to help understand how they work so effectively, which leads into the multiple deep dream image manipulation projects and studies. The paper then discusses another image manipulation technique using deep convolutional generative adversarial networks (DCGANs), how it works and the different research projects using them.

An art historians interpretation on the artistic and historic merits of deep dreaming images is included to augment the subjective nature of critiquing the produced images. This work discusses the implementation with the Inception V3 and training with a custom dataset, as well as tests and parameter changes done in

the different feature layers, gradient step sizes, and other variables. Results are discussed on whether this project succeeded in producing holiday themed images, and how changes to the dataset, training time, and deep dream image manipulator network parameters improved this performance. Issues and problems the team faced, including switching projects away from object detection and tracking, and gaining access to powerful computational resources on Intel AI Dev cloud, and problems with fine-tuning the image manipulator networks are discussed. The future plans for DeepHoliday include vastly increasing the size and categories of the image classification dataset, and training on different known convolutional neural network architectures and DCGANs.

II. Hidden Layers within Convolutional Networks

Neural networks are made up of many different components when it comes to processing data and transferring accurate measures. When focusing on hidden layers within the network, it is often confusing to figure out what they are actually doing. There are many questions that computer scientists ask when dealing with a topic such as deep learning, machine learning, deep dream, and the majority of them can be answered when working with these networks. There consists a series of layers in the neural networks where they are key factors. Layers within these networks basically create the structure or architecture of the neural network that allows data to be recognized, created, and even shared with other supporting networks within a system. Many experts say that hidden layers within the neural networks are the key factors as to why our computers and units are the way they are today.

Complex networks have shaped systems and important units in our government, as well as major companies in which allowed them to surpass many other organization around the world. Convolutional neural networks are only getting smarter over time. They are trained to learn through working and using examples from the real world. Many researchers have pointed out that deep convolutional networks are computing progressively more powerful invariants as depth increases, but relations with networks weights and nonlinearities are complex. [4]. This is where the process of hidden layers come into play. Hidden layers make up the complex functions by adding features to images that come from examples and use them to complete the next task. So with all these hidden layers involved in these complex systems, our networks are able to make accurate guesses and later provide even better results as data is collected. The layers take in the information that is given, breaks it all down, goes through every single piece of data, and outputs an accurate representation of what it is. So when an image is passed into the network, it is taken in as a whole, passes through all of the different hidden layers which take each piece of data and processes it, then is outputted with its correctly matched image. Afterwards, comes smaller processes within those hidden layers such as linearization. This is the process known in machine learning that takes the dimensions of an image and rescales it such that the images can be identified more easily and the output can be more accurate [4].

Recently, a program was launched that allowed users to intervene with the hidden layers within the network which, in turn, allowed them to change whatever they wanted to get different outcomes. Daniel Smilkov and Shan Carter were the creators of the program known as the Neural Network Playground [11]. With all the available options for both deep and machine learning, producers have made today's networks more conservative [4]. Neural networks would not be able to operate without the hidden layers inside them. They are the key ingredients when recognizing an image that is passed through. Without layers, the program couldnt be trained since there is little to be learned. There is just no possible way to create something or describe something with having

hidden layers.

III. Previous work on Image Manipulation with Deep Learning

A. Understanding Neural Networks Through Deep Visualization

Deep Dream itself is a Convolutional Neural Network that has been shown a large amount of images, usually all of a specific type, in the hope that the network would be able to understand exactly what makes whatever object an object while hopefully ignoring the unnecessary details that the training images would still present. While this has produced several strange images that are more akin to an acid trip than something like an actual object, this has also produced new tools and procedures in order to better understand the process that the network is undergoing and possibly improve its output.

This project focuses on these tools. The first of these tools is one that visualizes the activations produced on each layer of a trained (Convolutional Neural Network) as it processes an image or video, [18] while the second tool enables visualizing features at each layer of a Deep Neural Network via regularized optimization in image space. [18] The first of these is achieved by displaying a large amount of grayscale images of a specific size for whatever layer that has been selected; such that, for a layer of size 256x13x13, the tool would display 256 different 13x13 images arbitrarily laid out in a 16x16 image grid. Each of these small images would display what parts of the channel were activated. The second tool provides a visualization method for many of the patterns that were picked up by the network, which culminate in many of the swirl-like patterns that are notable on images processed by Deep Dream.

B. Visualising an Image Classification Model

There have been a few techniques which have been made to analyze and understand the processes done in image-based convolutional neural networks [10]; though these were designed for use in image classification networks, the ideas present have been used for image manipulation networks as well. One such technique for visualising what goes on in image-based convolutional neural networks is to have the network generate an image which maximises the class score [10] such that the resulting image should be representative of whatever classes of interest were learned by the convolutional network. This results in an output of a swirly-like image with edges and patterns that sort of resemble the type of image it represents in real life, akin to the swirls found in images processed by Deep Dream.

C. Understanding Deep Image Representations by Inverting Them

One such procedure developed in the wake of Deep Dream was an attempt to reconstruct the original image from one that has been modified by a Convolutional Neural Network. Deep Dream is making an effort to better understand how such networks work. This process makes use of the optimization of an objective function with gradient descent [3] which is capable of recovering certain lower-level aspects of the image that were removed by the earlier network. The one key factor that deep dream is trying to do is to take the image and completely invert it. Which in a sense is a bit confusing, however what this allows deep dream to do is recognize every aspect of the image. The problem of inverting representations, particularly CNN-based ones, is related to the problem of inverting neural networks, which has received significant attention in the past.[3]

D. Inverting Visual Representations with Convolutional Networks

Another project which dives into the idea of inverting an image-based convolutional neural

network in order to see how it works managed to find that features from all layers of the network... preserve the precise colors and the rough position of objects in the image [1] such that the original image could be reconstructed with the data from any of these layers, though more information was lost with the deeper layers. The network they had to train to invert these images back to their original form was an up-convolutional neural network [1], which is a CNN that, instead of working with a bottom-up and feed-forward architecture [16], works with two bottom-up and one top-down sub-networks in order to improve its discriminative capability.
Thus, by inverting the image, it allows them to find out everything going on within the image, reconstruct it, and add or take away what they need to accurately. By deep dream reverting images, insights are obtained within the invariances for representation [3]. Within the process, there are many different cases of the images and the one that is most accurate to the original is chosen and then manipulated again. In the future, deep dream is trying to come up with newer parameters for more natural images.

E. Deep Convolutional Generative Adversarial Networks

Deep Convolutional Generative Adversarial Networks (DCGAN) are a framework for training generative parametric models [17] which have been used to create high quality images and was used to tackle the issue of Semantic Image Inpainting, or in simpler terms, image restoration with a neural network. This sort of framework works off of two networks, a generator and a discriminator, which are both trained on uncorrupted data and with the optimization of the specific loss function mentioned in the sourced paper [17]. The generator network maps a random vector that has been sampled from a prior distribution to the image space while the discriminator network discriminates between what the generator creates and the sampled images from when the network was trained. This leads to a more effective image generation due to both networks working against each other (hence, the adversarial title) and, as the generator works to create the new image, the discriminator ensures that the resulting image will be reasonable considering what the networks were trained off of and what the original image actually was (for example, on a network trained off of several portraits, the resulting image should look like another portrait instead of something unnatural).
While this has shown to be an effective network framework in the above example [17], this sort of network would not be ideal for our specific project due to the non-sequitur nature of the input images in comparison to the training images, such that the network would have no idea what to do with a picture of the Pittsburgh skyline when it was trained entirely on Christmas-themed images. Theoretically, however, it could be possible to run this sort of network after running an image through a more conventional Deep Dream-like network. Then, the DCGAN would try to complete the Holiday-themed images after the first network placed various swirls and colors that only partially resemble the theme they are supposed to achieve.

F. Image Manipulation with Perceptual Discriminators

Image manipulation has come a long way in creating realistic alterations, such that realism that has been added to an already realistic image has become more popular with society today. Scientists working on neural networks today are capable of altering images within networks. Perceptual losses and losses based on adversarial discriminators are the two main classes involved with image manipulations [13]. These classes have led to advancements in image processing. Special systems or architectures have been used to train dis-

criminator networks within the system to build their framework for the entirety of the networks [13]. Studies have recently proven systems undergo training for billions of images that are annotated with the presence or absence of a certain attribute, rather than based on aligned datasets containing image pairs, allowing them to alter images more accurately. [13]. If the network is able to be trained with inaccurate data and then adapt so it can recognize the inaccuracies and output accurate data, that will further the networks range and make the entire system more flexible [13]. Thus, allowing the programmers to add to their network without having to worry about certain restrictions.

G. Generative Visual Manipulation on the Natural Image Manifold

As is described in the work Generative Visual Manipulation on the Natural Image Manifold, the idea behind this project is to allow users to make changes to images while maintaining their natural look [19]. Image manipulation, although easy for humans to accomplish, becomes difficult when trying to maintain the natural look of the image. In the paper, the team describes how their goal is to create a program that provides a user with special editing tools that will output an image that falls under the natural manifold that is created using a generative adversarial neural network [19].

As the paper delves further into the difficulties of image manipulation, their process comes about as a series of steps involving the creation of a projection that can be edited by the program based on the original image. The process goes as follows: using a generative adversarial network, the program will project the original image on a dimensional latent vector representation [19]. This vector representation is the thing that is manipulated by the user with the tools provided by the program. Once the user has manipulated the image to their liking, the program makes changes to the original image that were made to the vector projection. Despite this logic, the group claims that the results leave much to be desired, with the main problem being that the program doesnt always maintain the images natural look [19].

This project is beneficial in its ideals in the case of the Deep Holiday project. In the Deep Holiday project, there must be some sense of realism in the resulting image, otherwise the project does not accomplish its goal of effectively putting a holiday theme on any given image.

IV. Connections Between Science and Art

For many years, art has been used as a platform to illustrate various perspectives in music, text, and–for the focus of this paper–images. Visual art, also known as imagery, is the ability to create an image-based representation of ones thought process. The artists, being the individuals behind the work, have amazed the world with spectacular paintings and drawings that have been stored as historical moments of cultural representation. Nowadays, with the growth of technology, artists have found new ways to express their visual art to the rest of the world. Erwin Panofsky, inventor of the science of identification for art history stated that, How we perceive an object, classify it, and interpret it (in relation to other objects and other images) is a complex question that may not be trendy to ask in art history today, but has everything to do with the origins of the discipline, and has particular relevance to the way art historians in pre-Modern fields often approach the objects they study [12]. In the statement above, Panofsky attempts to convey that human inspiration has been the known driving force behind the various art styles [12]. While most of the inspiration stems from environment factors, it is important to note that post modern technology such as machine learning is now comparable to what is considered human inspiration; however, an outstanding difference is the mere fact that the computer can only learn from the inputs it has obtained from the

user. This can only generate unique visuals based on what the computers set parameters were set to. Even so, computer technology can generate remarkably vivid visual effects which can be seen as a new art style. This is good from an art historical perspective since computer vision science, similar to human vision science, requires visual abilities in identifying images based on trained data acquired by the subject [12]. In relation to the project at hand, training a system in such a way that would output holiday-themed images can be perceived as an addition to the new wave of technological art styles. Some of the benefits of manipulating images via machine learning or training are increases in the various forms of arts which can lead to potential changes in cultural behavior or significant moments in our life.

V. Project Implementation

A. Network Architecture

The Holiday DeepDream model is implemented with the Keras Inception V3 convolutional network. Inception famously set the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14) [14]. This network was chosen due to its existing and successful Keras implementation for deep dream. Other examples of Keras networks used for deep dream image manipulation could not be found, so this was the best choice for network implementation.
. The Inception V3 network is is a variant of Inception V2 which adds BN-auxiliary, meaning that the fully connected auxiliary classifiers are batch-normalized as well as the convolutional layers[15]. The Inception V2 is an improvement on the Inception V1 where convolutions are factored into smaller convolution, such as factoring the 7x7 convolution into three 3x3 convolutions[15]. The Inception network consists of Inception modules stacked upon each other, with occasional max-pooling layers of stride 2 to halve the resolution of the grid [14]. An Inception module approximates an optimal local sparse structure in a convolutional vision network with readily available dense components [14]. The module analyzes the correlation statistics of the last layer and clusters them into groups of units with high correlation, which then form the units of the next layer and are connected to the units in the previous layer[14]. The full Inception V3 network is shown below in Fig. 1 [6]. A fully connected layer classifier layer with 1000 sigmoid outputs is included for training the model with the Holiday images dataset.

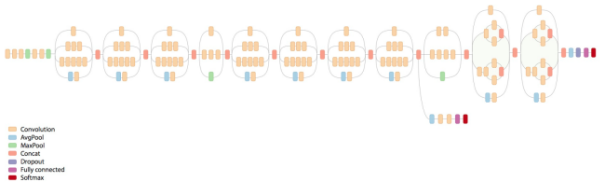

Fig. 1. Inception V3 architecture

B. Training and Testing Dataset

The dataset is split into 80 percent training and 20 percent testing folders, each of which contains 1000 labeled folders, with five different labeled images: santa for Santa Claus, trees for christmas trees, snowman for snowman, fireworks for fireworks, and menorahs for menorahs. The train folder contained 452 Santa Claus images, 310 christmas tree images, 467 snowman images, 471 menorah images, and 294 fireworks images for training. The validation folder contained 108 Santa Claus images, 76 christmas tree images, 108 snowman images, 111 menorah images, and 72 fireworks images for testing. The data was pulled from a list of urls from Google Images of the specific categories using JavaScript and Python scripts [8]. These images were chosen in order to carefully and concisely create a holiday only dataset, with a focus on Christmas due to its popularity, while also including Hanukkah, New Years Eve and the 4th of July.

C. Training and Testing DeepHoliday

The model was initialized with random weights to allow complete training on holiday images. The model was trained with a mini-batch size of 10, and two epochs, producing a training accuracy of 0.9990 percent and a testing accuracy of 0.9986 percent. The accuracy did not improve after two epochs, so that was the final decision for the number of epochs. The model was compiled with binary-cross entropy loss function, an RMSprop optimizer, and set for accuracy, parameters recommended for Inception V3 [11]. The trained weights and biases file was uploaded to a second Inception V3, this time fully convolutional with no fully connected final classification layer. Code from a Keras Deep Dream project developed by the Keras team and altered and provided to us by Alan Jamieson was adapted for our image manipulation. There was a features setting that was utilized to vary the activation strength of 4 different mixed feature layers: mixed2, mixed3, mixed4, and mixed 5. Other settings included the step size for gradient ascent, the number of scales allowed, and the allowed ratio between these scales. Sample input images included a city skyline of Pittsburgh, the Baltimore Penn Station, academic buildings and student housing at St. Marys College of Maryland (SMCM), and the Math and Computer Science Department at SMCM. By setting all mixed 10 layer features to zero except one, that layers effect on the original image, shown in Figure 2, could be distinguished, as shown in Figures 3 through 6.

VI. Results

Through many different tests on both the training and manipulating aspects of the project, the most optimal results found produced the following manipulations, shown in Figures 7 through 10.

From what can be seen in the results, the network produces manipulations that seem to replicate a combination of objects in the training data. Upon close inspection, one may no-

Fig. 2. Original Pittsburgh image

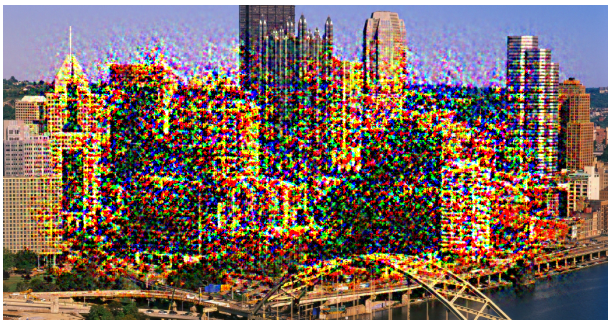

Fig. 3. mixed2 layer activated

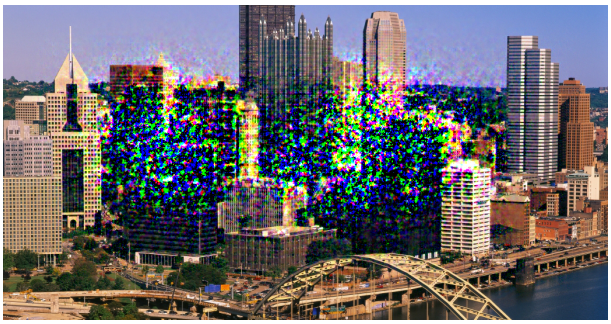

Fig. 4. mixed3 layer activated

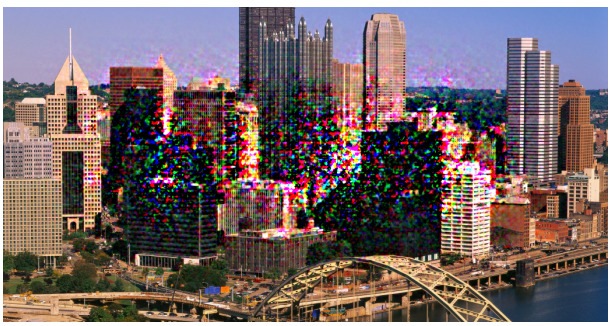

Fig. 5. mixed4 layer activated

Fig. 6. mixed5 layer activated

Fig. 7. St. Mary's Math and CS faculty original

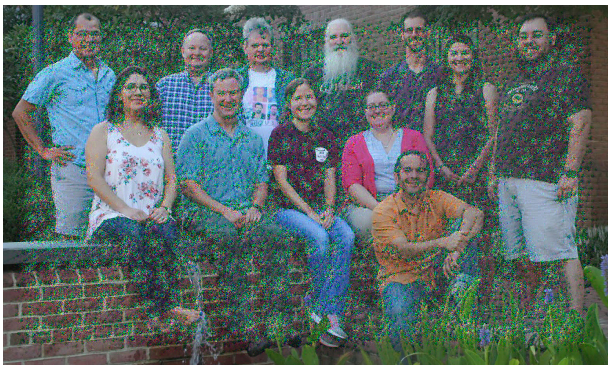

Fig. 8. St. Mary's Math and CS faculty manipulated

Fig. 9. St. Mary's Crescents housing original

Fig. 10. St. Mary's Crescents housing manipulated

tice that the fuzzy inclusions in the image seem to form into small ornament-like balls. Upon a less focused view, the manipulations seem to attempt to turn objects in the images into things like trees through the use of these focused ornament-like balls (as can be seen in the first of the two manipulations shown above). As part of the dataset, there are many images that included snow, despite snowy environments not being a specific focus of the dataset used. As can be seen in the second of the two results shown above, the network almost seems to produce a snowfall-like environment, while still using the ball-like focuses to do so. Due to the results produced by the use of these ball-like concentrations of fuzz, much of the testing went into attempting to bring about these concentrations. The reason for seeking these concentrated points is that the network would seem to use them as tools to show objects in an image. Through the many tests done on changes made to the network, the best results proved to be the ones that included these fuzz concentrations. Without these concentrations, there was no clear manipulation being made to the image (except for color). Many of the variants tried would produce large clouds of unrecognizable shapes, so it was determined that the best results were just enough to show meaningful manipulation while also not having too much. How the fuzzy concentrations come about and what they are meant to be are not precisely known due to the nature of this

kind of deep learning, but it would seem as though they appear in attempts to show/create some kind of object (perhaps trees in the first example and snow in the second). Such results may not be deemed a success in terms of enhancing the look of an image, but could be considered successful in terms of making an image appear more holiday related. These relations to holiday related items are shown in the somewhat unclear objects created as well as the colors added to the images, as they are mostly consistent with those of the dataset.

A. Path to the Final Results

As was mentioned earlier, the main focus of testing was to make changes that would encourage the networks use of the fuzzy concentration points. With the two main aspects of the project being the training and the manipulating, most of the testing and variations revolved around performing manipulations with saved weights. Changes made to the training file had significantly less impact on the outcome compared to the manipulation file. This is likely due to the fact that the dataset used had many empty folders to try to compare to each other, leaving the accuracy very high from the beginning of training. This was done because the program required 1000 folders, empty or not, in order to function properly, and using more data than what was used in this project proved to be complicated to obtain as well as increase the runtime significantly. The prominent number of empty files were likely the cause of such consistently high accuracy when training. This also meant that there was either no practical way to know when images in the dataset were properly used for training, or that the accuracy was just that high and not going to be significantly impacted by the images in the dataset.

Regardless of the reason for the high accuracy, most of the attention went to the manipulation file. With the manipulation file many different changes were made. In this file, there were four activation layers to make changes to. As testing showed, the first two of which mixed2 and mixed3 would influence the amount of color and large unrecognizable shapes would show up in the manipulation. The last two mixed4 and mixed5 proved to be of more interest for this project. Both of them would show more recognizable shape manipulations, and they would include less color than the first two layers. With this in mind, mixed4 was set to the highest value as it created the most clear shapes, followed shortly by mixed5 which made recognizable, but blurry shapes. mixed2 and mixed3 were both set rather low in value in order to maintain the colors they added to the image without them completely overtaking the manipulations. To get the best results, the balance between how light and how defined the changes were had to be found. From there, significant testing went into finding the correct step size, octave scale, iterations, and max loss. As it were, the number of iterations was set to 30 in order to find the perfect time to allow changes to be made while also not allowing the changes to be too impactful. What also helped with this was the max loss. Any loss above 1 would typically result in a completely damaged image, so 1 was determined to be the best value to give max loss. The most significant variable after those was the step size. When the step size was too high, the number of iterations would never be met, and the changes would be too deep in some places while leaving others unchanged. When the step size was too small, the number of iterations would have to be increased, and the parts of the image where most of the image was manipulated would be darkened beyond recognition. The testing showed that the current settings for each variable were the ideal ones in all of the pictures that were used for testing, though that is not to say that these are the ideal settings for all images.

VII. Issues and Concerns

During the course of the project, many issues had to be dealt with. These issues ranged

from major compatibility issues to major topic difficulties. The section below covers many aspects of the project that were problematic and what was done to solve said issues.

A. Environments

One issue included requiring a specific environment on the team's account for Intel's AI Development Cloud Computing Service in order to run the project without the trainer producing NaN (Not A Number) weights, and producing black output images. The environment needed keras, tensorflow, and many other specific versions of packages and dependencies to run the project successfully. The projects python code needed to be entered into an ipython shell rather than running the files conventionally. This requires users to access the specific environment in order to run the project, making collaborative work more difficult.

B. Previous Project: Object Tracking Through Occlusion

he first topic for this project did not involve image manipulation. Instead, the original topic was object tracking through occlusion, but that turned out to be a much larger problem than the team was able to handle within the given time. One of the major difficulties of this topic was finding a place to start. With few having done such a task as tracking objects through occlusion using deep learning, there is not much content to be found on the internet that would help with accomplishing such a thing. Once a piece of code was found that seemed it would fit the needs of the project, it was chosen for use as a baseline to get started.

One of the more prominent projects involving object tracking through deep learning is GOTURN, which is a neural network that tracks objects after being trained offline at 100 frames per second [2]. The directions provided by the GOTURN creators [2] are ineffective in terms of getting the code to run without issue. Without proper instruction, it seemed impossible to get GOTURN to work at all, much less in the scope of the tracking through occlusion project. The time required for completing this project required a more suitable piece of code. Upon a further research, a project called Multiple Object Tracking System in Keras + (Detection Network - YOLO) [9] better fitted the needs of the object tracking through occlusion project. More interestingly, the code for that project seemed to be two programs combined into one (as its title would suggest). The way this program works is that it uses a program called YOLO (You Only Look Once) [7] in order to detect objects in any given image, then uses the results of that object detection in order to track the objects shown. YOLO [7] is an interesting deep learning program that provides capable instruction in terms of getting it to run properly, so that was not problematic.

That is not to say it was not without problems. Even running the code on an I7-7700 CPU and a GTX 1080 GPU would take hours for a single epoch to complete, and the code recommended that 100 epochs are completed (about a week or more for total completion of 100 epochs). For the purpose of testing ideas and techniques, training through a single epoch proved to be acceptable. Unfortunately, the main problem with the Keras Tracking System + YOLO [9] code is that the format of the videos and accompanied files were not specified effectively enough for the object tracking portion of the code. This kind of issue is of great importance as it prevents further progress with the tracking objects through occlusion project.

The plan that was settled upon to alleviate this was to have a portion of the group attempt to find the correct format while the rest of the group would attempt to work with YOLO [7] to try and prepare it for training to recognize partial objects (with the idea that it would hopefully be able to recognize a part of an object through occlusion). It was

possible to set up YOLO [7] in such a way that allowed it to recognize partial faces, but it showed little-to-no success in detecting objects effectively through occlusion. Even if the object detection code were able to detect through occlusion, there was still the concern of object tracking through occlusion. Without working code that could use the object detection to track the detected objects, there was no more that could be done with the Keras Tracking System + YOLO [9] project. With little other option, the topic of the project was changed to image manipulation to lower the difficulty to reasonable levels within the remaining time.

VIII. Future Work

Future work includes using a larger dataset of christmas images, involving more categories such as angels, pumpkins, valentines, and other holiday imagery, as well as increasing the number of images in each category from the hundreds to the thousands, and possibly the tens of thousands. Plans also include using other well known deep convolutional networks such as GoogLeNet and VGGNet [15], to test their ability for effective deep dream image manipulation. Another task would involve using DCGANs trained on a holiday dataset to provide festive and holiday themed manipulations on areas of other non-holiday themed images.

IX. CONCLUSIONS

The work on DeepHoliday has demonstrated the first known instance of performing deep dream image manipulation on non-holiday images to make them more festive, training the Inception V3 network on a custom holiday images dataset. The project showed how using deep dreaming to manipulate the image by gradient ascent produces various altercations, ranging from creepy to more appealing results. This work is important in that it demonstrates a specific application of deep dream for the holiday season, and continued work on this project could produce significantly more profound results. The results produced by DeepHoliday are indeed more holiday themed than the non holiday input images, though this evaluation is inherently subjective, as different groups of people have different definitions on a holiday image. This work also highlights the uncanny effectiveness of deep convolutional neural networks, and how visualizing the activations of these hidden layers assist in our understanding. Although the initial object tracking project failed when data for the tracker could not be entered in the suitable format, the switch to image manipulation in the last week was a successful adjustment. The DeepHoliday image manipulator was built and tested promptly to give reasonable results, and the team shifted their research focus entirely toward deep dream, CNNs, and DCGANs. Difficulties arose in running the code without producing a black output image file, and efforts to run the code in ipython in a specific users environment were set to circumvent this problem. Overall, in spite of all these setbacks, the ability to completely change projects and produce a fully working deep learning solution for image manipulation indicates that this project was a success.

References

[1] A. Dosovitskiy, T. Brox: Inverting Visual Representations with Convolutional Networks, Computer Vision and Pattern Recognition, 4829-4837 (2016)

[2] D. Held, S. Thrun, S. Savarese: Learning to Track at 100 FPS with Deep Regression Networks, European Conference on Computer Vision, (2017).

[3] A. Mahendran, A. Vedaldi: Understanding Deep Image Representations by Inverting Them, Institute of Electrical and Electronics Engineers. 5188-5196. (2015).

[4] S. Mallat: Understanding Deep Convolutional Networks, Philosophical Transactions of the Royal Society A, 1471-2962 (2016).

[5] A. Mordvintsev, C. Olah, M. Tyka: Inceptionism: Going Deeper into Neural Networks. Google AI Blog, https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html (Updated June 17, 2018).

[6] T. Mulc: Inception modules: explained and implemented, Hack Till Dawn https://hacktilldawn.com/2016/09/25/inception-modules-explained-and-implemented/ (September 25, 2016)

[7] J. Redmon, A. Farhadi: YOLO: Real-Time Object Detection https://pjreddie.com/darknet/yolo/ (updated 2018).

[8] A. Rosebrock: How to create a deep learning dataset using Google Images, pyImageResearch https://www.pyimagesearch.com/2017/12/04/how-to-create-a-deep-learning-dataset-using-google-images/ (December 4th, 2017).

[9] K. Sharma. Multiple Object Tracking System in Keras + (Detection Network - YOLO) https://github.com/kshitiz38/object-tracking (Updated June 3, 2018).

[10] K. Simonyan, A. Vedaldi, A. Zisserman: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Computer Vision and Pattern Recognition (2014)

[11] D. Smilkov, S. Carter: Tinker With a Neural Network Right Here in Your Browser. Dont Worry, You Cant Break It. We Promise. **https://playground.tensorflow.org/activation=tanhbatchSize=10dataset=circleregDataset=reg-planelearningRate=0.03regularizationRate=0noise=0networkShape=4,2seed=0.54280showTestData=falsediscretize=falsepercTrainData=50x=truey=truexTimesY=falsexSquared=falseySquared=falsecosX=falsesinX=falsecosY=falsesinY=falsecollectStats=falseproblem=classificationinitZero=falsehideText=false** (updated March 9, 2017).

[12] E. L. Spratt: Dream Formulations and Deep Neural Networks: Humanistic Themes in the Iconology of the Machine-Learned Image, Kunsttexte.de, (2017).

[13] D. Sungatullina, E. Zakharov, D. Ulyanov, V. Lempitsky: Image Manipulation with Perceptual Discriminators, European Conference on Computer Vision, (2018).

[14] C. Szegedy , W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich: Going Deeper with Convolutions, Computer Vision and Pattern Recognition, (2015).

[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna: Rethinking the Inception Architecture for Computer Vision, Computer Vision and Pattern Recognition, (2016).

[16] C. Xu, J. Yang, H. Lai, J. Gao, L. Shen, S. Yan: UP-CNN: Un-pooling Augmented Convolutional Neural Network: Pattern Recognition Letters (August, 2017)

[17] R. A. Yeh, C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, M. N. Do: Semantic Image Inpainting with Deep Generative Models, Computer Vision and Pattern Recognition, (2017).

[18] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, H. Lipson: Understanding Neural Networks Through Deep Visualization, International Conference on Machine Learning, (2015).

[19] J. Zhu, P. Krahenbuhl, E. Shechtman, A. A. Efros: Generative Visual Manipulation on the Natural Image Manifold, European Conference on Computer Vision, (2018).