Deep Learning Image Recognition: Analysis and Evaluation

Alex Salman
Department of Computer Science
UC Santa Cruz
Santa Cruz, CA 95064
aalsalma@ucsc.edu

Abstract

Convolutional Neural Networks (CNNs) recognize images in a way that can not be proofed. We can call CNNs as Black Boxes. For recognition, we do not know if CNNs are using the shape in the image or the texture. It is believed that CNNs should recognize images as humans do. The shape is more important than the texture, size, and color of the objects. So in this paper, I am analyzing and evaluating the work of (Geirhos et al., 2019)[1]. I am evaluation their findings and analyzing their outcome.

1 Introduction

When reading research papers in the field of Deep Learning, I noticed that there is a gap between claimed results of some papers' findings and the actual ones people produce when reimplementing them. Further, there is a need to analyze and evaluate the findings of research papers that could be used in a variety of applications/ academic research. With my interest in learning about how CNNs recognize images [2], I picked to reimplement this research paper, ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness [1], to explore how image recognition is happening in Black Boxs, CNNs, as well as how CNNs are biased towards texture when trained on ImageNet (IN) dataset.

CNNs are commonly thought to recognize objects by learning increasingly complex representations of object shapes. They combine low-level features (e.g. edges) to increasingly complex shapes (such as wheels, car windows) until the object (e.g. car) can be readily classified. High-level units appear to learn representations of shapes occurring in natural images. CNNs have been proposed as computational models of human shape perception. They develop a so-called "shape bias" just like children, i.e. that object shape is more important than color for object classification. It is well-known that object shape is the single most important cue for human object recognition, much more than other cues like size or texture.

However, some recent studies suggest a more important role of image texture! CNNs can still classify texturized images perfectly well, even if the global shape structure is completely destroyed. Standard CNNs are bad at recognizing object sketches where object shapes are preserved yet all texture cues are missing. It seems that local textures indeed provide sufficient information about object classes. So, IN object recognition could, in principle, be achieved through texture recognition alone.

2 Methods

To make a comparison between how humans recognize objects and how CNNs do, the authors of the paper performed nine comprehensive and careful psychophysical experiments comparing humans

Preprint. Under review.

(a) Texture image
81.4% **Indian elephant**
10.3% indri
8.2% black swan

(b) Content image
71.1% **tabby cat**
17.3% grey fox
3.3% Siamese cat

(c) Texture-shape cue conflict
63.9% **Indian elephant**
26.4% indri
9.6% black swan

Figure 1: A cat with an elephant texture is an elephant to CNNs, and still a cat to humans.

name	training	fine-tuning	top-1 IN accuracy (%)	top-5 IN accuracy (%)	Pascal VOC mAP50 (%)
vanilla ResNet	IN	-	76.13	92.86	70.7
	SIN	-	60.18	82.62	70.6
	SIN+IN	-	74.59	92.14	74.0
Shape-ResNet	SIN+IN	IN	**76.72**	**93.28**	**75.1**

Table 1: Fine-tuning is not making good improvement in accuracy.

against CNNs on exactly the same images [3]. These experiments provide behavioral evidence in favor of the texture hypothesis. (Figure 1)

Since the experiment has been conducted using humans and lab tools, I only worked on experiments that involve using different datasets with different deep learning models. However, I have presented the paper in another class to get feedback on the paper as described in the section below:

2.1 Paper presentation

In order to better understand the work and how good the authors' findings are, I presented the paper in a detailed form in a Deep Learning class [4]. The main feedback I received is that the fine-tuning done is not making a good improvement in the accuracy nor the object detection performance [5], as shown in (Table 1).

Further, the finding of the paper that highlights the importance of Content image is not robust considering the metrics achieved in the paper. Could the texture be more important?

2.2 Re-implementation

As suggested by the author, I followed the implementation recommended in the paper GitHub README section [1]. The sequence is as shown in (Figure 2).

```
# get softmax output
softmax_output = SomeCNN(input_image) # replace with your favourite CNN

# convert to numpy
softmax_output_numpy = SomeConversionToNumpy(softmax_output) # replace with conversion

# create mapping
mapping = probabilities_to_decision.ImageNetProbabilitiesTo16ClassesMapping()

# obtain decision
decision_from_16_classes = mapping.probabilities_to_decision(softmax_output_numpy)
```

Figure 2: re-implementation snippet

```
model = load_model(model_A) # change to different model as desired
im = pilimage.open('style-transfer-preprocessed/'+x)
test_transforms = transforms.Compose([transforms.ToTensor()])
image_tensor = test_transforms(im).float()
image_tensor = image_tensor.unsqueeze_(0)
out1 = model(image_tensor)
out_smax_values = F.softmax(out1, dim=1).detach().cpu().numpy()
```

Figure 3: Softmax Function

```
# create mlapping
mapping = probabilities_to_decision.ImageNetProbabilitiesTo16ClassesMapping()
# obtain decision
out_smax_values = np.array([j for j in out_smax_values[0][:]])
decision_from_16_classes = mapping.probabilities_to_decision(out_smax_values)
```

Figure 4: Getting Prediction

Starting from using already trained models, I used the three models used in the paper, Resnet50, VGG16, and Alexnet. With all the models trained on Stylized ImageNet (SIN), a synthetic dataset that includes 16 categories sets in aspects of Original, Greyscale, Silhouette, Edges, and Texture that applied AdaIN style transfer to IN images [6][7][8][9]. Further, I used Resnet50 with a combination of IN and SIN. And another one with IN and SIN but fine-tuned.

The piece of code below in (Figure 3) is my implementation to use the softmax function. From torchvision, I used transforms to covert images I want models to predict to float tensors [10]. Then I used torch.nn.functional function softmax to get the output.

Before sending the softmax output to do the mapping, I converted the data structure to NumPy array as it is the required data structure by probabilities to decision function as shown in the code in (Figure 4).

3 Results

I have conducted some experiments to compare the metrics found in the paper and the ones I got from using different models. (Figure 5) shows the performance of Resnet50 trained in IN and SIN and the humans' performance on the same sets of images.

On Resnet50 trained on SIN, I tested sets of edges using Cat, Airplane, and Boat images. The results respectively came as follow: Cat: 50%, Airplane: 60%, Boat: 50%. And that's close to the presented in the edges style in (Figure 5).

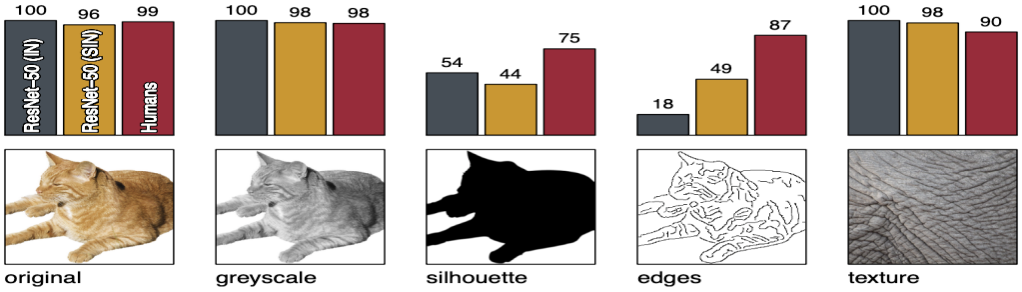

Figure 5: Different Datasets

Table 2: Experiment 1

Model/Categories	Cat	Airplane	Boat
Resnet50 trained on SIN	50%	60%	50%
Alexnet trained on SIN	30%	40%	30%
VGG-16 trained on SIN	10%	90%	80%

Table 3: Experiment 2

Model/Categories	Keyboard	Chair	Knife
Resnet50 trained on SIN	0%	100%	60%
Alexnet trained on SIN	0%	100%	80%
VGG-16 trained on SIN	0%	100%	90%

On Alexnet and VGG-16 that are trained on SIN, I got the results for Alexnet: Cat 30%, Airplane: 40%, Boat: 30%, and VGG-16: Cat 10%, Airplane: 90%, Boat: 80%. That shows a drastic difference among the models on the same datasets. Results of Experiment 1 are shown in (Table 2).

Besides having differences in between the models, a set of Silhouettes shows a drastic correct prediction percentage among other categories I used such as Keyboard, Chair, and Knife. Results of Experiment 2 are shown in (Table 3).

That indicates the shape of the keyboard is too general to be correctly predicted; however the shape of the chair is kind of unique which makes the prediction pretty accurate.

In addition to using Edges and Silhouettes datasets, I tested a Style-transfer-preprocessed dataset (Cats shape with a sample of 80 images) and got some of the results in (Figures 6 and 7). The predictions of the models get bad when using synthetic images created from shape and texture. In (Figure 6), the shape is a cat and the texture from the left is truck, elephant, and bottles. (Figure 7) makes it clear that the bias is towards texture when the picture prediction is a dog when the shape is cat and the texture is dog's.

Last but not least, using a fine-tuned trained model, the prediction percentage has decreased for the same test set that I used in (Figure 6 and Figure7). For more test samples of this dataset, please see all 80 images run output on the following jupyter notebook run in the project repository: `https://github.com/alexsalman/NeuralComputation/blob/main/texture_vs_shape.ipynb`

Looking at (Figure 8), the model is pretty biased even though it's fine-tuned. The example on the left predicts chair which is the texture of the image and not the shape, cat. Same case for the example on the right side. The prediction is truck, following the texture!

Figure 6: Experiment 3

```
cat7-dog3.png
Using the ResNet50 architecture.
Prediction is:  dog
```

Figure 7: Experiment 3

```
cat1-chair2.png
Using the ResNet50 architecture.
Prediction is:  chair
```

```
cat2-truck3.png
Using the ResNet50 architecture.
Prediction is:  truck
```

Figure 8: Experiment 3 - ResNet50 trained on SIN and IN then fine-tuned on IN Cat: 17.5% (80 images)

4 Challenges

4.1 Project Software/Hardware Setup

As my plan was to use the school's remote GPU through a VPN, I have run into some issues in installing needed software on the remote server. Permission issues made me change to plan B which is using Google Cloud Platform Instance with an NVIDIA Tesla V100 to run the paper code on the cloud. Furthermore, I had to install Jupyter Notebook on the gcloud instance for easy use of the remote storage as well as code run and prediction presentation.

4.2 Softmax Conversion

Applying softmax was a challenging part for me in the reimplementation process. With a little background knowledge in CNNs, I had to learn how this function works and how to implement it and work with the input and obtain output for it [11][12].

5 Conclusion

Although different experiments with different images give varying results, the main pieces of insight and observation are still the same. The bias against object shape. It is accurate that the trained on IN + SIN models work better in predicting the images that have less texture; however, fine-tuning on this

combination of the dataset doesn't work as reported in the paper. Sometimes, it gives worse results than without fine-tuning.

That keeps the ambiguity of how the Black Box, CNNs, actually recognize images. And the prevailing observation is still the same, the prediction is better when there is texture in the image regardless of how good the shape is.

Acknowledgement

This term project would not be done without the help of the class instructor and a couple of classmates who helped in setting up my work environment with a GPU.

References

[1] Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F., Brendel, W. (2019, January 14). ImageNet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. Retrieved December 06, 2021, from https://arxiv.org/abs/1811.12231

[2] Traore, B., Kamsu-Foguem, B., Tangara, F. (2018, October 12). Deep Convolution Neural Network for Image Recognition. Retrieved December 06, 2021, from https://www.sciencedirect.com/science/article/pii/S1574954118302140

[3] Bethgelab. (n.d.). Bethgelab/model-vs-human: Benchmark your model on out-of-distribution datasets with carefully collected human comparison data (neurips 2021 oral). Retrieved December 06, 2021, from https://github.com/bethgelab/model-vs-human

[4] Salman, A. (2021, October 28). ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. Retrieved December 06, 2021, from https://docs.google.com/presentation/d/15GJf7MG9zXD8FQi652qGjqW4KAXcxDSnL10x5wJmCSA/editslide=id.p

[5] Jordan, J. (2018, December 05). Hyperparameter tuning for machine learning models. Retrieved December 06, 2021, from https://www.jeremyjordan.me/hyperparameter-tuning/

[6] Arbitrary style transfer in real-time - arxiv. (n.d.). Retrieved December 6, 2021, from https://arxiv.org/pdf/1703.06868.pdf

[7] Welcome to pytorch tutorials. (n.d.). Retrieved December 06, 2021, from https://pytorch.org/tutorials/

[8] Rgeirhos. (n.d.). Rgeirhos/stylized-imagenet: Code to create stylized-ImageNet, a stylized version of standard ImageNet (ICLR 2019 oral). Retrieved December 06, 2021, from https://github.com/rgeirhos/Stylized-ImageNet

[9] Leongatys. (2020, June 02). Pytorchneuralstyletransfer/NeuralStyleTransfer.ipynb at master · Leongatys/Pytorchneuralstyletransfer. Retrieved December 06, 2021, from https://github.com/leongatys/PytorchNeuralStyleTransfer/blob/master/NeuralStyleTransfer.ipynb

[10] Python torchvision.models.resnet50() examples. (n.d.). Retrieved December 06, 2021, from https://www.programcreek.com/python/example/108013/torchvision.models.resnet50

[11] Automatic differentiation in Pytorch - OpenReview. (n.d.). Retrieved December 6, 2021, from https://openreview.net/pdf?id=BJJsrmfCZ

[12] Dietz, M. (2017, May 09). Understand deep residual networks - a simple, modular learning framework that has redefined... Retrieved December 06, 2021, from https://medium.com/@waya.ai/deep-residual-learning-9610bb62c355