Model Inference and Possible Extrapolations

Andrea Covre
7643
Georgia Tech
andrea.covre@gatech.edu

Jake Hopkins
4644
Georgia Tech
jhopkins39@gatech.edu

Connor Reitz
4644
Georgia Tech
creitz@gatech.edu

Abstract

Over the past years, a large amount of research, innovation, and thought has gone into creating deep neural networks that achieve a high level of performance. However, model theft is a topic of recent popularity: the possibility of replicating an existing model with minimal assumptions and thought, thereby eliminating both intellectual effort and monetary costs associated with this long process. We will explore whether inferred models can attain similar performance to their stolen counterparts and what consequences this could impose on this field. Our approach to simulate this process is to create an image classifier using AI-generated images. Our findings indicate that it is possible to create models with similar performances. However, accomplishing this feat requires much more time and many more resources than originally thought.

1. Introduction

This paper is an in-depth exploration of model inference: is it possible to develop a high-performing learning model through inferring either data or labels from a pre-existing model architecture? This has many implications in the machine learning field - for example, model theft is a security risk to pre-existing models where an attacker could replicate a model at a low-cost and create their own for free use and personal gain. On a more positive side, ethical, low-cost labeling of datasets could be achieved through successful model inference using free-use models for actors who are strapped for cash (research, startups). As such, we aim to demonstrate whether model inference is able to result in the creation of models to the current standard of industry acceptable models.

This paper explores one main scenario and expands from there:

- Inferring a model through mining data examples from a pre-existing generative model and subsequently training our own models with different architectures and measuring performance

2. Related Works

Knockoff Nets [6] have been studied and implemented in previous research, which attempt to steal functionality of victim models. From a computer vision perspective, knockoff models were able to be created by providing images to retrieve labels, using these image label pairs to train the knockoff. Previous work in this domain is similar to what we aim to achieve, but we differ in our approach in that we are planning to measure performance between different architectures, and we plan on also attempting the converse approach: inferring from labels instead of the data through a generative model.

NLP Model Extraction [4] has shown to be effective as well - even using random word input to steal functionality from existing NLP models. This work also shows example defenses against these attacks, such as removing outliers for non-sensical inputs and watermarking some outputs to purposefully be incorrect. We will not delve into defenses here, but random input is an interesting point to consider in model inference.

GNN Model Extraction [8] has been researched and shown to be effective as well, and as a whole demonstrates that a wide variety of model architectures are able to be stolen.

2.1. What We Do Differently

Our goal is to provide concrete proof that somebody can start with nothing, generate synthetic data, and use this data to help with tasks in the real world or simulate already existing models. This isn't necessarily a novel idea, but we aim to provide ways to improve results with this synthetic data. Our different methods to increasing accuracies and results with the generated data is what sets this paper apart.

3. Technical Approach

This section will describe how we attempt to perform model inference, demonstrate its capabilities, and aim to evaluate ourselves at the task. We will compare ourselves to a baseline and ultimately, the criterion will be used to show success or failure.

3.1. Baseline

There are some simple baseline solutions to 'model inference' that we are working on and researching in this paper.

A baseline to check synthetic or labeled data would be using human classification power. They could go through and evaluate the created images by hand. This however is infeasible, especially when looking at huge datasets of size greater than 100,000 or 1 million data points.

A baseline for the problem of generating your own model that mimics another would be to start from scratch and design and train a model after collecting a good labeled data set.

This is how it is normally done in the deep learning field and we're hoping we can speed up this process by using AI generated synthetic data. For many deep learning tasks, collecting data (especially by hand) can be a significant part and take research teams weeks to do properly. It would be amazing for us to show that AI data is good enough for real world applications; we're hoping our Proof of Concept (POC) will show this.

3.2. Method

The first step will be to gather synthetic data. We have written a script to mine AI-created images from Craiyon (formerly the Dalle-mini). With this script, we create 60,000 images across the 10 classes meaning 6,000 images for each class.

Now that we have our dataset we can begin to play and experiment with it to see what we can achieve. We call this new dataset Fake-CIFAR-10. Now we will attempt two different tasks.

For the first task, there are 4 parts. Training model with real data and validating with Real. Training with real and validating with fake. Training with fake and validating with fake. Training with fake and validating with real. For each of these tasks we create a confusion matrix to see how the model performs. The confusion matrix shows for each ground-truth label: how many are classified as that label. Here we refer to the specific dataset that we are using to validate's labels as the ground-truth labels.

There will be high accuracy on the first case because that is why it is a public, well-known model - it can perform well.

High accuracy on the second case would mean that our synthetic data (at least to computer standards) is high quality data. It is almost a way to evaluate the image generation software. We expect this to be high because image creation evaluates the created images and only returns the created image when it can be classified as such with a high enough probability. As we move forward with AI generated images, using a separate AI to evaluate these may be a good course of action and help with the image generation.

For the third case we are going to use the same model architecture, but untrained. We will then train the model with Fake-CIFAR-10 data and validate it with the real data. We will train using industry standard gradient descent and using pytorch to implement the steps. Success will show how a model trained using synthetic data can be used in the real world to make real world predictions. This is a proof of concept of effectively "stealing a model." While we use the same architecture this demonstrates that it would be theoretically possible to generate fake data to train a model that performs as accurately as models created by other researchers and industry players.

Finally, for the final case, we train and validate the model with Fake-CIFAR-10. This again is another metric for evaluating the dataset and the successes of the image generator.

For the second task, we aim at increasing the accuracy of our model trained on fake data and validated with real data. We're aiming for a high accuracy because this would represent the ability to use synthetic data for real-world applications. We try a variety of different ways ranging from dropping bad data or adding in additional and different AI-generated images.

Additionally, we look at creating a convolutional neural net to create a classifier of real images versus AI-generated images. This task stems from having already created a classifier for the real and the fake images.

An example step forward could be to be able to classify more items. We could pick 10 more objects, that are separate from the CIFAR classifications, quickly generate images and have your own classifier for these 10 objects. Start-ups could use this for cheap AI or certain actors (including students) could use this to help get around expensive software.

3.3. Evaluation Criteria

Initially, our main evaluation criteria is the confusion matrix seen below 4. The confusion matrix shows how successful the current net we're using is at accurately classifying. We use this as the main criterion because correct classification is the main goal of the nets we are analyzing.

As we work more on this we will also look at recall, precision, and F2 score to be able to identify success and failure of the nets.

Case 3 of task 1 and task 2 will the most interesting to look at. High accuracy ($> 90\%$ or $> 95\%$) in case 3 will mean that synthetic data is good enough to be truly be used in the real world.

High accuracy ($> 90\%$ or $> 95\%$) in task 2 will mean that a model can effectively be ”stolen.” This is because starting with unlabeled and random data we were able to develop a model that will show the same results and classifications.

4. Data

We attempt to create a dataset using public AI-generative models that is comparable to the CIFAR-10 dataset.

4.1. Collection Process

We generated our own synthetics dataset through Crayon [1] which is a public AI model that allows any user to generate a set of 9 images from any text prompt.

To leverage this public model and compose our synthetics dataset we first reverse-engineered the website to expose the API that sends the text prompt to the backed model and then returns the generated images. The Crayon API is a simple HTTP request that after 40 seconds (on average) returns a list of 9 elements where each element is a base 64 string encoding of the 256x256 generated image. Given this information, we wrote a script that uses Crayon’s API with one of the 10 classes of the CIFAR-10 dataset (e.g. *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck*) as text prompt. We were able to parallelize the requests and mine around 30 images per second with a new request every 0.3 seconds (a faster rate would have caused us to run into HTTP 429 Too Many Requests errors which would have decreased the throughput). In a little bit over half an hour, we were able to generate a “free” synthetic dataset of the same size as CIFAR-10 (60,000 images total, 6,000 images per class). This fast and “cost-free” process enables us to automatically generate a dataset of any object/class by only changing the target classes used as text prompts in Crayon’s API.

4.2. Preprocessing

After generating and collecting our synthetic dataset, we had to further process it to have a final result similar to CIFAR-10, therefore we had to prune down the number of images to 6,000 per class, resize each image from 256x256 to 32x32, and finally randomly split the dataset in training and validation sets, in a ratio of 5:1 just as CIFAR-10.

The last step to finalize our artificial dataset was to calculate its mean and standard deviation to normalize it during the training and validation phases and to also compare it with the real CIFAR-10 dataset.

CIFAR-10 Set	RGB Mean	RGB STD
Real	0.4919, 0.4827, 0.4472	0.2470, 0.2434, 0.2616
Fake	0.4689, 0.4952, 0.4226	0.2463, 0.2468, 0.2799

Table 1. The channel-wise mean and standard deviation between the two datasets is very comparable and the small differences are expected.

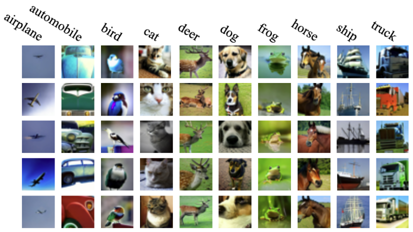


Figure 1. Examples of Crayon generated images (synthetic data).

5. Results

In our preliminary approach to this problem, we mostly focused on synthetic data rather than labels, therefore mining and composing a synthetic dataset that resembles the real CIFAR-10 dataset [5], testing the accuracy of a pre-trained ResNet56 against this synthetic data, and then actually training the ResNet56 with such data.

5.1. Baseline metrics

To evaluate how feasible using synthetic data is, we want to first measure the accuracy of ResNet52 pre-trained on CIFAR-10 (with 94.37% Top-1 Acc. and 99.83% Top-5 Acc.) [2] when predicting our synthetic data. Then we want to look at the confusion matrix to see if there is any particular class of synthetic data that the baseline model cannot perform well on.

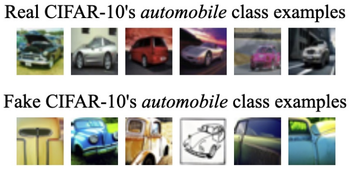
After this initial evaluation we want to try to invert the roles, therefore train ResNet52 on the synthetic data (with the same hyperparameters as the pre-trained model), and evaluate its performance while predicting the data from the real CIFAR-10 dataset through raw accuracy and confusion matrix.

5.2. Synthetic data results and analysis

When we evaluated ResNet56 trained on Real CIFAR-10 dataset with Real and Fake CIFAR-10 test sets we obtained 94.06% and 95.81% accuracy respectively. We suspect that the validation against the Fake test set resulted in a higher accuracy because the AI model that generated the images tends to accentuate the prompt/labeled object identifying features.

The confusion matrix in Fig.4 shows that the Real-on-Real performance is overall very good besides occasionally confusing cats for dogs and dogs for cats. While Fig.5

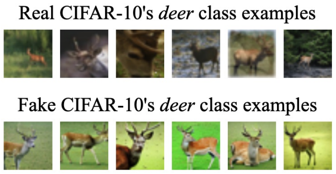
shows that the Real-on-Fake performance is mostly better than Real-on-Real's one, except for *automobile* class which is quite often confused with the *truck* and *airplane* classes. This is caused by the fact that the Fake dataset's *automobile* class is not a good, generalized representation of what an automobile looks like the distribution of examples generated by the AI model is skewed toward a specific category of automobiles (vintage automobiles), while the Real dataset contains a wider variety of automobiles that range from vintage to modern models as shown in Fig.2.


Real CIFAR-10's *automobile* class examples

Fake CIFAR-10's *automobile* class examples

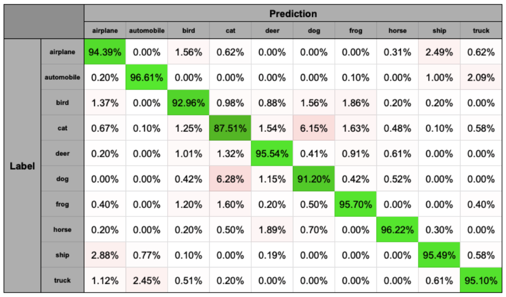
Figure 2. Comparison of Real and Fake *automobile* examples.

Another observation worth mentioning is the suspiciously high accuracy (100%) on the *deer* class in the Real-on-Fake experiment which we can notice in the corresponding confusion matrix in Fig.5. This perfect accuracy is caused again by the low variety of *deer* examples in the Fake dataset. As shown in Fig.3, the Real dataset depicts deer with different backgrounds, at different distances, and from different angles, while the Fake dataset has very easy examples where the background is quite repetitive, and the deer always have very similar dimensions and positioning.


Real CIFAR-10's *deer* class examples

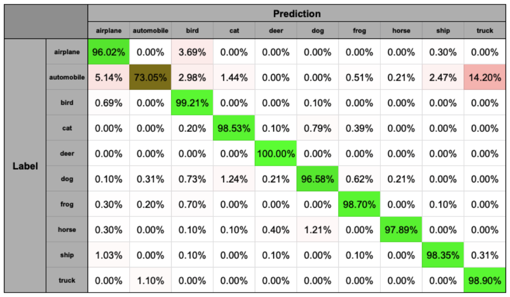
Fake CIFAR-10's *deer* class examples

Figure 3. Comparison of Real and Fake *deer* examples.



		Prediction									
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Label	airplane	94.39%	0.00%	1.56%	0.62%	0.00%	0.00%	0.00%	0.31%	2.49%	0.62%
	automobile	0.20%	96.61%	0.00%	0.00%	0.00%	0.00%	0.10%	0.00%	1.00%	2.09%
	bird	1.37%	0.00%	92.96%	0.98%	0.88%	1.56%	1.86%	0.20%	0.20%	0.00%
	cat	0.67%	0.10%	1.25%	87.51%	1.54%	6.15%	1.63%	0.48%	0.10%	0.58%
	deer	0.20%	0.00%	1.01%	1.32%	95.54%	0.41%	0.91%	0.61%	0.00%	0.00%
	dog	0.00%	0.00%	0.42%	6.28%	1.15%	91.20%	0.42%	0.52%	0.00%	0.00%
	frog	0.40%	0.00%	1.20%	1.60%	0.20%	0.50%	95.70%	0.00%	0.00%	0.40%
	horse	0.20%	0.00%	0.20%	0.50%	1.89%	0.70%	0.00%	96.22%	0.30%	0.00%
	ship	2.88%	0.77%	0.10%	0.00%	0.19%	0.00%	0.00%	0.00%	95.49%	0.58%
	truck	1.12%	2.45%	0.51%	0.20%	0.00%	0.00%	0.00%	0.00%	0.61%	95.10%

Figure 4. Confusion matrix of ResNet56 trained with Real CIFAR-10 and validated against Real CIFAR-10.



		Prediction									
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Label	airplane	96.02%	0.00%	3.69%	0.00%	0.00%	0.00%	0.00%	0.00%	0.30%	0.00%
	automobile	5.14%	73.05%	2.98%	1.44%	0.00%	0.00%	0.51%	0.21%	2.47%	14.20%
	bird	0.69%	0.00%	99.21%	0.00%	0.00%	0.10%	0.00%	0.00%	0.00%	0.00%
	cat	0.00%	0.00%	0.20%	98.53%	0.10%	0.79%	0.39%	0.00%	0.00%	0.00%
	deer	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	dog	0.10%	0.31%	0.73%	1.24%	0.21%	96.58%	0.62%	0.21%	0.00%	0.00%
	frog	0.30%	0.20%	0.70%	0.00%	0.00%	0.00%	98.70%	0.00%	0.10%	0.00%
	horse	0.30%	0.00%	0.10%	0.10%	0.40%	1.21%	0.00%	97.89%	0.00%	0.00%
	ship	1.03%	0.00%	0.10%	0.00%	0.10%	0.00%	0.10%	0.00%	98.35%	0.31%
	truck	0.00%	1.10%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	98.90%

Figure 5. Confusion matrix of ResNet56 trained with Real CIFAR-10 and validated against Fake CIFAR-10.

We can conclude then, that ResNet56 trained on the Real CIFAR-10 can classify very well a synthetic dataset generated from a generative model.

Now lets, investigate if the opposite is also true. We trained ResNet56 from scratch with the same hyper-parameters (listed in Table 2) as the pre-trained model we used earlier.

Then we ran the same experiments by evaluating the Fake-on-Real and Fake-on-Fake performances for which we obtained respectively 43.05% and 98.91% accuracy. These results corroborate our hypothesis that the Fake dataset tends to overly display key identifying features of the classes, therefore our model overly relies on these features to classify images. When the model has to predict images from the Real dataset, it does not find these key features as prominently as in the Fake training set, hence it fails to properly identify the correct class.

Epochs	Batch size	Optimizer	LR	Momentum	Weight decay	Nesterov
200	256	SGD	0.1	0.9	0.0005	Yes

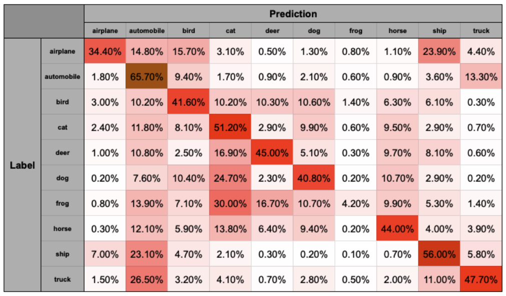
Table 2. Hyper-parameters used to train both the pre-trained model and our own model.

Training set	Acc. on Real CIFAR	Acc. on Fake CIFAR
Real CIFAR	94.06%	95.81%
Fake CIFAR	43.05%	98.91%

Table 3. Accuracies obtained with ResNet56 model trained with Real or Fake data and validated on Real or Fake data.

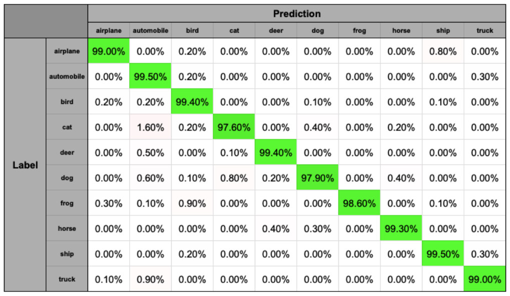
5.3. Alternative model

We also ran the same experiments using the RepVGG-A2 architecture to have an alternative model to compare our results to. We decided to use RepVGG-A2 because its pre-trained model has higher accuracy on CIFAR-10 and because while ResNet56 only has 0.86M parameters, RepVGG-A2 has over 26M parameters [2]. However, our experiments showed that the RepVGG-A2 model trained



		Prediction									
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Label	airplane	34.40%	14.80%	15.70%	3.10%	0.50%	1.30%	0.80%	1.10%	23.90%	4.40%
	automobile	1.80%	65.70%	9.40%	1.70%	0.90%	2.10%	0.60%	0.90%	3.60%	13.30%
	bird	3.00%	10.20%	41.60%	10.20%	10.30%	10.60%	1.40%	6.30%	6.10%	0.30%
	cat	2.40%	11.80%	8.10%	51.20%	2.90%	9.90%	0.60%	9.50%	2.90%	0.70%
	deer	1.00%	10.80%	2.50%	16.90%	45.00%	5.10%	0.30%	9.70%	8.10%	0.60%
	dog	0.20%	7.60%	10.40%	24.70%	2.30%	40.80%	0.20%	10.70%	2.90%	0.20%
	frog	0.80%	13.90%	7.10%	30.00%	16.70%	10.70%	4.20%	9.90%	5.30%	1.40%
	horse	0.30%	12.10%	5.90%	13.80%	6.40%	9.40%	0.20%	44.00%	4.00%	3.90%
	ship	7.00%	23.10%	4.70%	2.10%	0.30%	0.20%	0.10%	0.70%	56.00%	5.80%
	truck	1.50%	26.50%	3.20%	4.10%	0.70%	2.80%	0.50%	2.00%	11.00%	47.70%

Figure 6. Confusion matrix of ResNet56 trained with Fake CIFAR-10 and validated against Real CIFAR-10.



		Prediction									
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Label	airplane	99.00%	0.00%	0.20%	0.00%	0.00%	0.00%	0.00%	0.00%	0.80%	0.00%
	automobile	0.00%	99.50%	0.20%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.30%
	bird	0.20%	0.20%	99.40%	0.00%	0.00%	0.10%	0.00%	0.00%	0.10%	0.00%
	cat	0.00%	1.60%	0.20%	97.60%	0.00%	0.40%	0.00%	0.20%	0.00%	0.00%
	deer	0.00%	0.50%	0.00%	0.10%	99.40%	0.00%	0.00%	0.00%	0.00%	0.00%
	dog	0.00%	0.60%	0.10%	0.80%	0.20%	97.90%	0.00%	0.40%	0.00%	0.00%
	frog	0.30%	0.10%	0.90%	0.00%	0.00%	0.00%	98.60%	0.00%	0.10%	0.00%
	horse	0.00%	0.00%	0.00%	0.00%	0.40%	0.30%	0.00%	99.30%	0.00%	0.00%
	ship	0.00%	0.00%	0.20%	0.00%	0.00%	0.00%	0.00%	0.00%	99.50%	0.30%
	truck	0.10%	0.90%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.00%

Figure 7. Confusion matrix of ResNet56 trained with Fake CIFAR-10 and validated against Fake CIFAR-10.

on the real data did perform slightly better, but the model trained on the fake data performed slightly worse on the fake test dataset, and significantly worse on the real test dataset where the accuracy decreased by over 8% as shown in Table 5. Since the RepVGG-A2 models performed worse (especially the model trained on the fake data), and it takes significantly more time to train, we decided to move forward with only the ResNet56 architecture.

Training set	Acc. on Real CIFAR	Acc. on Fake CIFAR
Real CIFAR	95.14%	96.42%
Fake CIFAR	34.57%	97.88%

Table 4. Accuracies obtained with RepVGG-A2 model trained with Real or Fake data and validated on Real or Fake data.

6. Differentiating between Real and Fake data

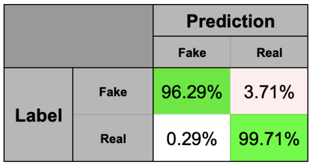
Since ResNet56 trained on the Fake data cannot perform well on the Real test data, we wanted to explore if a modified version of the architecture could at least accurately differentiate between the images from the Fake dataset and the Real one. We expect the model to perform decently well on this task since from the previous experiments it's pretty clear that there are some characteristics in the Real dataset that are not displayed in the Fake one and viceversa.

We explored the following two scenarios.

6.1. ResNet56 with 2 classes

In this scenario we changed the last fully connected layer of ResNet56 so that the size of the network's output equals 2, effectively classifying 2 classes: Fake dataset, and Real dataset. We trained the network with the same hyper-parameters listed in Table 2, and using a custom dataset composed by both the Real and Fake ones merged together, and where the labels where changed to 0 for the fake images and 1 for the real ones.

This model performed relatively well, with an accuracy of 98%. In Fiugre 8 we can see that the model tends to misclassify fake images for the real ones.



		Prediction	
		Fake	Real
Label	Fake	96.29%	3.71%
	Real	0.29%	99.71%

Figure 8. Confusion matrix of 2-classes ResNet56 trained to differentiate between Fake CIFAR-10 and Real CIFAR-10.

6.2. ResNet56 with 20 classes

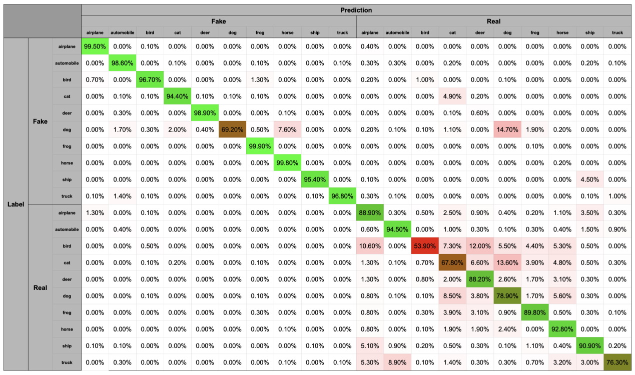
In this second scenario, we followed the same steps as in the case with only 2 classes, but this time we wanted to see if the model could classify at the same time both the dataset of origin (Fake or Real) and the appropriate CIFAR-10 class. Therefore the modified model has output size equal to 20 (with labels: real_dog, fake_dog, real_cat, fake_cat, etc.).

This model performed worse than the 2-classes one, with an accuracy of only 88.56%. In Figure 9 we can see that the model particularly struggles in specifically differentiating the classes of the real images, which was unexpected since the 2-classes model classified better the real images compared to the fake ones.

Another peculiar result is that both the 2-classes and 20-classes ResNet models mistake fake images for real ones 10 times more often than they mistake real image for fake ones.

Modified ResNet Accuracies	
2-classes	98.00%
20-classes	88.56%

Table 5. Accuracies obtained with the modified ResNet56 models aimed to differentiate between the Real and Fake datasets.



			Prediction																			
			Fake										Real									
			airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Label	Fake	airplane	99.50%	0.00%	0.10%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.40%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
		automobile	0.00%	98.60%	0.00%	0.10%	0.00%	0.00%	0.10%	0.00%	0.00%	0.10%	0.30%	0.30%	0.00%	0.20%	0.00%	0.00%	0.00%	0.00%	0.20%	0.10%
		bird	0.70%	0.00%	96.70%	0.00%	0.00%	0.00%	1.30%	0.00%	0.00%	0.00%	0.20%	0.00%	1.00%	0.00%	0.00%	0.10%	0.00%	0.00%	0.00%	0.00%
		cat	0.00%	0.10%	0.10%	94.40%	0.10%	0.10%	0.10%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	4.90%	0.20%	0.00%	0.00%	0.00%	0.00%	0.00%
		deer	0.00%	0.30%	0.00%	0.00%	98.90%	0.00%	0.00%	0.10%	0.00%	0.00%	0.00%	0.00%	0.00%	0.10%	0.60%	0.00%	0.00%	0.00%	0.00%	0.00%
		dog	0.00%	1.70%	0.30%	2.00%	0.40%	69.20%	0.50%	7.60%	0.00%	0.00%	0.20%	0.10%	0.10%	1.10%	0.00%	14.70%	1.90%	0.20%	0.00%	0.00%
		frog	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.90%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.10%	0.00%	0.00%	0.00%
		horse	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.80%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.20%	0.00%	0.00%
		ship	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	95.40%	0.00%	0.10%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	4.50%	0.00%
		truck	0.10%	1.40%	0.10%	0.00%	0.00%	0.00%	0.00%	0.00%	0.10%	96.80%	0.30%	0.10%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.10%	1.00%
	Real	airplane	1.30%	0.00%	0.10%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	88.90%	0.30%	0.50%	2.50%	0.90%	0.40%	0.20%	1.10%	3.50%	0.30%
		automobile	0.00%	0.40%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.60%	94.50%	0.00%	1.00%	0.30%	0.10%	0.30%	0.40%	1.50%	0.90%
		bird	0.00%	0.00%	0.50%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	10.60%	0.00%	53.90%	7.30%	12.00%	5.50%	4.40%	5.30%	0.50%	0.00%
		cat	0.00%	0.00%	0.10%	0.20%	0.00%	0.00%	0.10%	0.00%	0.00%	0.00%	1.30%	0.10%	0.70%	67.80%	6.60%	13.60%	3.90%	4.80%	0.50%	0.30%
		deer	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.30%	0.00%	0.80%	2.00%	88.20%	2.60%	1.70%	3.10%	0.30%	0.00%
		dog	0.00%	0.00%	0.10%	0.00%	0.00%	0.00%	0.10%	0.00%	0.00%	0.00%	0.80%	0.10%	0.10%	8.50%	3.80%	78.90%	1.70%	5.60%	0.30%	0.00%
		frog	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.30%	0.00%	0.00%	0.00%	0.80%	0.00%	0.30%	3.90%	3.10%	0.90%	89.80%	0.50%	0.30%	0.10%
		horse	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.10%	0.00%	0.00%	0.80%	0.00%	0.10%	1.90%	1.90%	2.40%	0.00%	92.80%	0.00%	0.00%
		ship	0.10%	0.10%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.10%	0.00%	5.10%	0.90%	0.20%	0.50%	0.30%	0.10%	1.10%	0.40%	90.90%	0.20%
		truck	0.00%	0.30%	0.00%	0.00%	0.00%	0.00%	0.00%	0.10%	0.00%	0.10%	5.30%	8.90%	0.10%	1.40%	0.30%	0.30%	0.70%	3.20%	3.00%	76.30%

Figure 9. Confusion matrix of 20-classes ResNet56 trained to differentiate between Fake and Real CIFAR-10, as well as the CIFAR-10 classes.

7. Experimentation on Further Improving Fake-on-Real Performance

As noted above, fake generated data performs poorly (43.05% accuracy) on real world testing examples. This undermines a core positive use-case of model inference wherein cash-strapped actors care able to employ free-use generative models to cheaply generate datasets. As such, we have explored different methods in an attempt to increase performance in this area of our model.

7.1. Data Dropout to Optimize Training Data

Here we will explore a data dropout method as described in ”Data Dropout: Optimizing Training Data for Convolutional Neural Networks” [9] and ”Understanding Black-box Predictions via Influence Functions” [3], aiming to reduce error by eliminating a subset of images in the training data that are harmful in the overall learning process. Summarized, the proposed algorithm uses a two-round training approach that calculates the influence on loss over the training set, eliminates data points classified as most harmful, and finally retrains the model using the newly optimized dataset.

The reason for considering this approach is that data created by generative models has a high chance to be noisy for any given label. Thus, eliminating outliers that are not necessarily representative of the label would seem to be beneficial for performance.

We implemented this approach to generate predictions on the 500 images that have the most harmful influence on the loss function, drop a certain subset of these images, and retrain the model. The following demonstrates results from differing numbers of the bad images dropped, using the same hyper-parameters described in Table 2.

These results do not follow our initial hypothesis; the more bad images that are dropped from the training set, the worse the accuracy on real test data becomes.

% of Bad Img Dropped	Acc. on Real CIFAR
100%	29.9%
25%	36.52%
10%	38.47%
0%	43.05%

Table 6. Effects on dropping a percentage of 500 images identified as harmful in training on the model

There are a number of potential reasons why this could be the case:

Number of outliers. There could be significantly less outliers than hypothesized leading to a larger removal images that could be classified as the ”most harmful” but still are helpful to training - which seems to follow the way our results have presented themselves. This also implies number of data points is more important than diminishing quality of data points.

Limitation of resources. For influence calculations it is suggested to have the number of s-test calculations to average times the recursion depth equal the training data set size. However, the tier of Google’s online web-service (GCE VM) used did not have enough memory to perform calculations with these suggested parameters. It is possible that with these constraints the resulting images classified as most harmful were not as precise as necessary to accurately remove bad images.

Misleading Classification. One interesting result of the influence on loss calculation is that almost 75% of the harmful images identified were in a subsection containing just 10% of the overall training data: a majority of the harmful images identified came from one or two classes. This could be the result of either misleading samples generated for a label or due to some other phenomenon - however, it is not likely that this behavior is intended.

It is difficult to tell exactly what causes this data dropout method to result in lower performance with a Fake-on-Real model, but further research with improved resources could pave the way into yielding more promising results.

7.2. Different AI-Generated Data

The main idea behind this approach is to improve the accuracy of our net by adding new data created by a different generative AI. Craiyon was originally picked due to the ease of mining the images (we were able to get 10s of thousands of images in a reasonable amount of time). Many of the other popular models, like OpenAI’s Dalle-2, make it much harder to mine a sufficient amount of images. We decided to use CompVis’s Stable-Diffusion-v1-4 [7]. They had an API allowing us to directly call the image generation onto whatever device we chose. However, even with a Google Cloud VM, it still took too long. We were only able to generate about 5,000 images (500 for each class). While it isn't enough to train a brand new model on, it is enough information to try different POCs that could lead to more substantive results down the line.

A visual analysis of these newly generated images tells us that some of these images look better and some of the images look worse than the Craiyon-generated images. This sent mixed signals, but the hope was that the better appealing images would perform better on all tasks. The visual analysis also showed us that we need to be careful with how we prompt the AI to generate these images. If no style was specified, CompVis would generate pictures in weird styles and formats. Because of this we had to specify "realistic photo of [class]". Even with these modifiers, "realistic photo of deer" could generate an image like Fig.10:

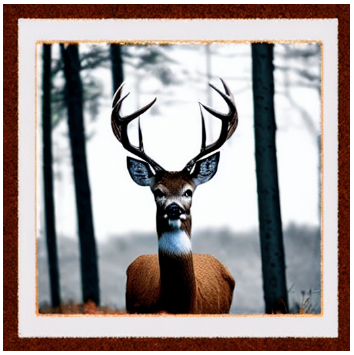
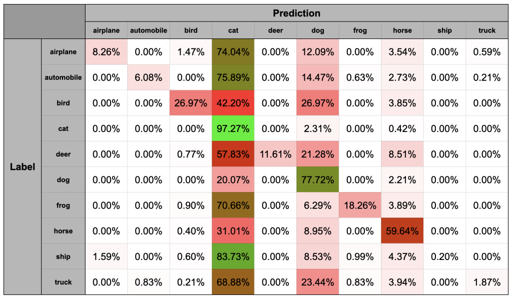


Figure 10. Deer Generated by CompVis with prompt "Realistic photo of deer"

After modifying the images to be the right size and normalizing the data set, we then performed some experiments on the data. We let the original ResNet56 attempt to classify these images and we had our model trained on fake data attempt to classify the images. Originally, we were hoping that both these tests would produce good results (as measured by high accuracy in the classification scores) and from there we would tweak our model to include these images and potentially, in the future, use these images to train a model.

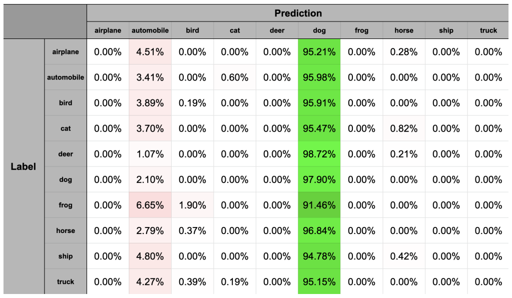
However, to our surprise, the results were not good. The real ResNet56 was unable to accurately classify the new images. It was slightly better than random guessing coming out with an accuracy of 33%. Our neural net trained on fake data performed terribly, classifying everything as a dog or an automobile and giving an accuracy of around 11% as seen in fig.11 and fig.12.

We think that the reason for this lack of results could come from how the images are generated. Both models use text-to-image and stable-diffusion to generate the images. There is a difference in how the models were trained to perform so differently for each of them. For example, the datasets the images are pulled from could be different. i.e. Craiyon could have used images much more similar to the CIFAR-10 dataset allowing them to work together better.



		Prediction									
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Label	airplane	8.26%	0.00%	1.47%	74.04%	0.00%	12.09%	0.00%	3.54%	0.00%	0.59%
	automobile	0.00%	6.08%	0.00%	75.89%	0.00%	14.47%	0.63%	2.73%	0.00%	0.21%
	bird	0.00%	0.00%	26.97%	42.20%	0.00%	26.97%	0.00%	3.85%	0.00%	0.00%
	cat	0.00%	0.00%	0.00%	97.27%	0.00%	2.31%	0.00%	0.42%	0.00%	0.00%
	deer	0.00%	0.00%	0.77%	57.83%	11.61%	21.28%	0.00%	8.51%	0.00%	0.00%
	dog	0.00%	0.00%	0.00%	20.07%	0.00%	77.72%	0.00%	2.21%	0.00%	0.00%
	frog	0.00%	0.00%	0.90%	70.66%	0.00%	6.29%	18.26%	3.89%	0.00%	0.00%
	horse	0.00%	0.00%	0.40%	31.01%	0.00%	8.95%	0.00%	59.64%	0.00%	0.00%
	ship	1.59%	0.00%	0.60%	83.73%	0.00%	8.53%	0.99%	4.37%	0.20%	0.00%
	truck	0.00%	0.83%	0.21%	68.88%	0.00%	23.44%	0.83%	3.94%	0.00%	1.87%

Figure 11. Confusion Matrix for real net attempting to classifying CompVis's data



		Prediction									
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Label	airplane	0.00%	4.51%	0.00%	0.00%	0.00%	95.21%	0.00%	0.28%	0.00%	0.00%
	automobile	0.00%	3.41%	0.00%	0.60%	0.00%	95.98%	0.00%	0.00%	0.00%	0.00%
	bird	0.00%	3.89%	0.19%	0.00%	0.00%	95.91%	0.00%	0.00%	0.00%	0.00%
	cat	0.00%	3.70%	0.00%	0.00%	0.00%	95.47%	0.00%	0.82%	0.00%	0.00%
	deer	0.00%	1.07%	0.00%	0.00%	0.00%	98.72%	0.00%	0.21%	0.00%	0.00%
	dog	0.00%	2.10%	0.00%	0.00%	0.00%	97.90%	0.00%	0.00%	0.00%	0.00%
	frog	0.00%	6.65%	1.90%	0.00%	0.00%	91.46%	0.00%	0.00%	0.00%	0.00%
	horse	0.00%	2.79%	0.37%	0.00%	0.00%	96.84%	0.00%	0.00%	0.00%	0.00%
	ship	0.00%	4.80%	0.00%	0.00%	0.00%	94.78%	0.00%	0.42%	0.00%	0.00%
	truck	0.00%	4.27%	0.39%	0.19%	0.00%	95.15%	0.00%	0.00%	0.00%	0.00%

Figure 12. Confusion Matrix for our fake net attempting to classifying CompVis's data

Another reason that these new images didn't work with models is that the CompVis images could be too weird to be compared with normal photos. Visual analysis already showed that the images came out in different designs and styles. If the "realistic photo" modifier didn't do enough to make the images seem like real images, then it would make sense that this dataset wouldn't be comparable.

Ultimately, generating different images with an AI was unable to increase overall accuracy due to a variety of possible reasons. A more in-depth analysis of CompVis's and Craiyon's image generation might provide more insight in the future.

8. Conclusion

Ultimately, our research provides promising evidence of the first steps towards model recreation with generated data. While 43% accuracy may not seem too high, the semblance of accurate results proves that it is theoretically possible. The different methods we attempted did not help improve this accuracy, but may provide insights into how to improve this accuracy in the future. The not so perfect results may be reassuring to some, that computers aren't on pace with humans quite yet.

Future development of our work can go a lot of different directions, but all of it, we believe would be focused on the data. Is there a better generative model that creates images that are more similar to real ones? Is there a better way to pick out the poorly generated images or ones that don't match the prompt perfectly? Going further into the future, if CIFAR-10 accuracies could be replicated with synthetic data, could CIFAR-100 be implemented with minimal changes?

The whole space is growing so quickly, and it has been very exciting to experiment like we did throughout the whole paper.

9. Group Contributions

9.1. Andrea Covre

Mined, processed and analyzed over 60,000 images from Crayon [1] to create our own custom dataset (Fake-CIFAR-10). Trained and tested the Real and Fake models with both the ResNet56 and RepVGG-A2 architectures. Created the merged Fake+Real dataset and modified, trained and tested ResNet56 to run the 2-classes and 20-classes experiments. Analysed experiments results and visualized the data.

9.2. Connor Reitz

Worked on exploring the data dropout option of improving Fake-on-Real accuracy through implementation of influence on loss to optimize the training data set. Analyzed these results and described them above. Wrote the abstract, introduction, and related works sections and worked on designing the initial experiment as well.

9.3. Jake Hopkins

Mined the 5,000 CompVis images using a GCE VM. Then, also ran the experiments with these images such as engineering the data, running them through ResNet56, and running them through our own model. Analyzed the results and wrote the section in the paper about them. Also wrote the methodology and conclusion sections above, formatted the final report, and helped design the experiment.

References

[1] Dayma Boris and Cuenca Pedro. Craiyon, formerly DALL-E mini. https://www.craiyon.com, 2022. 3, 8

[2] Chenyaofo. PyTorch CIFAR Models. https://github.com/chenyaofo/pytorch-cifar-models, 2021. 3, 4

[3] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions, 2017. 6

[4] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P. Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on sesame street! model extraction of bert-based apis, 2019. 1

[5] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 5(4):1, 2010. 3

[6] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models, 2018. 1

[7] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022. 6

[8] Yun Shen, Xinlei He, Yufei Han, and Yang Zhang. Model stealing attacks against inductive graph neural networks, 2021. 1

[9] Tianyang Wang, Jun Huan, and Bo Li. Data dropout: Optimizing training data for convolutional neural networks. *CoRR*, abs/1809.00193, 2018. 6