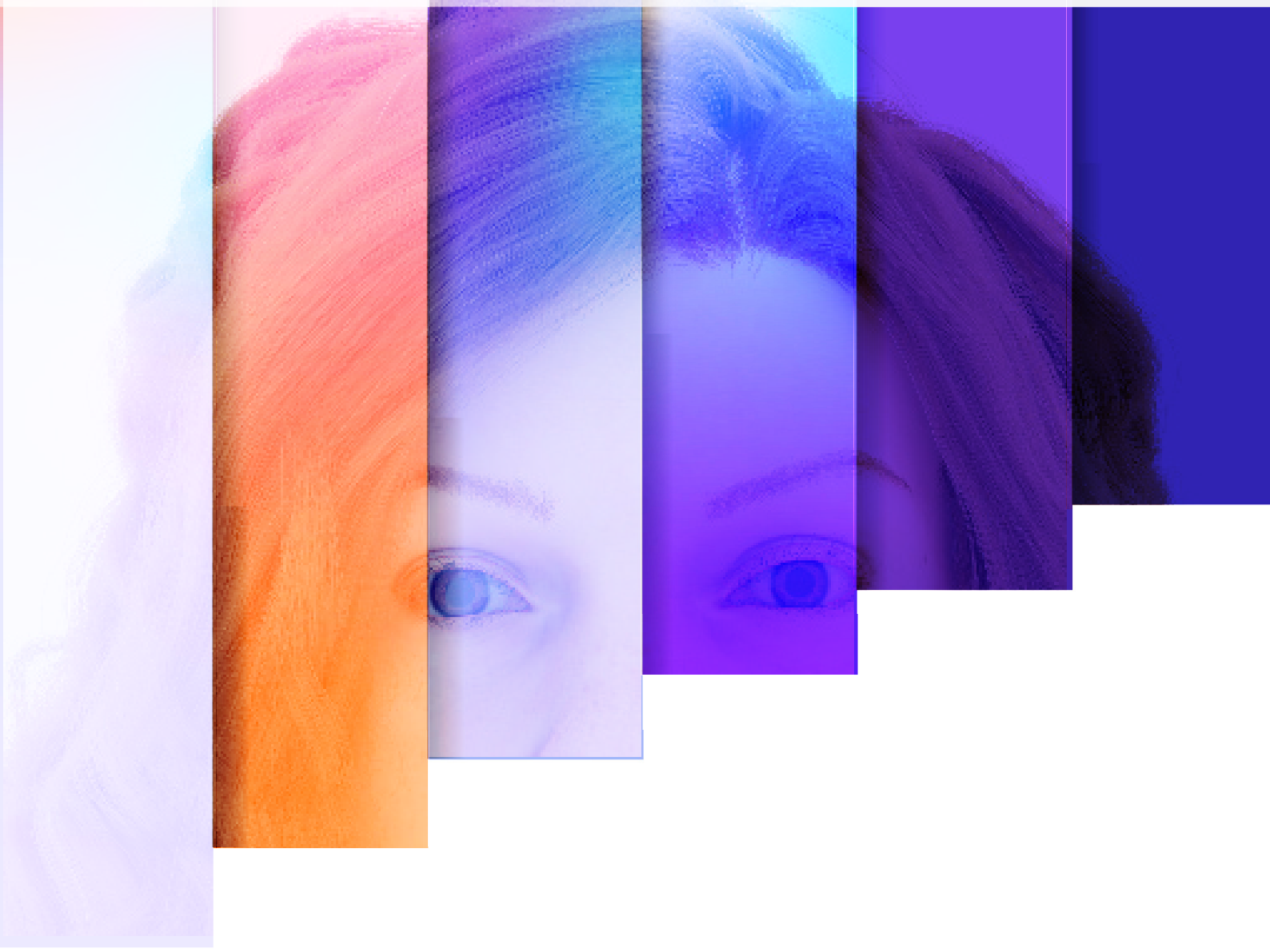
eBook

October 2022



How to Use Synthetic Data in 6 Easy Steps

datagen

Data-Centric Computer Vision is on the Rise

The proliferation of synthetic data is a key driver in the data-centric AI movement. As the data-centric movement garners strong momentum within the community, so has the use of synthetic data. Today, synthetic data is mature enough to complement, or at times replace, real-world data in training and test sets for computer vision applications.

For years, the machine learning community has prioritized the search for the best model over the hunt for better data. Practitioners speak of optimizing for hyperparameters and tweaking model architectures but rarely express the need for improving the existing dataset. That is starting to change.

"When a system isn't performing well, many teams instinctually try to improve the code. But for many practical applications, it's more effective instead to focus on improving the data," Andrew Ng shrewdly pointed out. He instead advocated for a data-centric approach, where the model is held constant and the data iteratively improved.

The data-centric approach is the preferred approach, especially for those working with small datasets. These practitioners are most likely to find more significant performance gains if they were to iteratively improve the data instead of the model. (Table 1)

	Steel Defect	Solar Panel	Surface Inspection
Baseline	76.2%	75.68%	85.05%
Model-Centric	+0% (76.2%)	+0.04% (75.72%	+0.00% (85.05%)
Data-Centric	+16.9% (93.1%)	+3.06% (78.74%)	+0.4% (85.45%)

Table 1: Andrew Ng compared three use cases at Landing.AI where data-centric approaches resulted in more significant performance gains than non-data-centric approaches. [Source]

A common strategy to improve a dataset is to increase its size and its diversity, which helps the model generalize across more scenarios and reduces the likelihood of overfitting.

Contrary to popular belief, there are more ways to improve a dataset than merely increasing its size. Instead, one can also refine its quality by ensuring label consistency for manually annotated data. A concrete way to guarantee consistency is to have multiple labelers labeling and agreeing (Figure 1) on all data points. Another method of ensuring label consistency is to remove ambiguous or subjective labels (e.g. the difference between a smile and a smirk is subjective when labeling an emotion) that might contribute to dirty training signals. Alas, attempts to improve data that are manually collected and annotated are often costly, operationally intensive, biased, and error-prone.

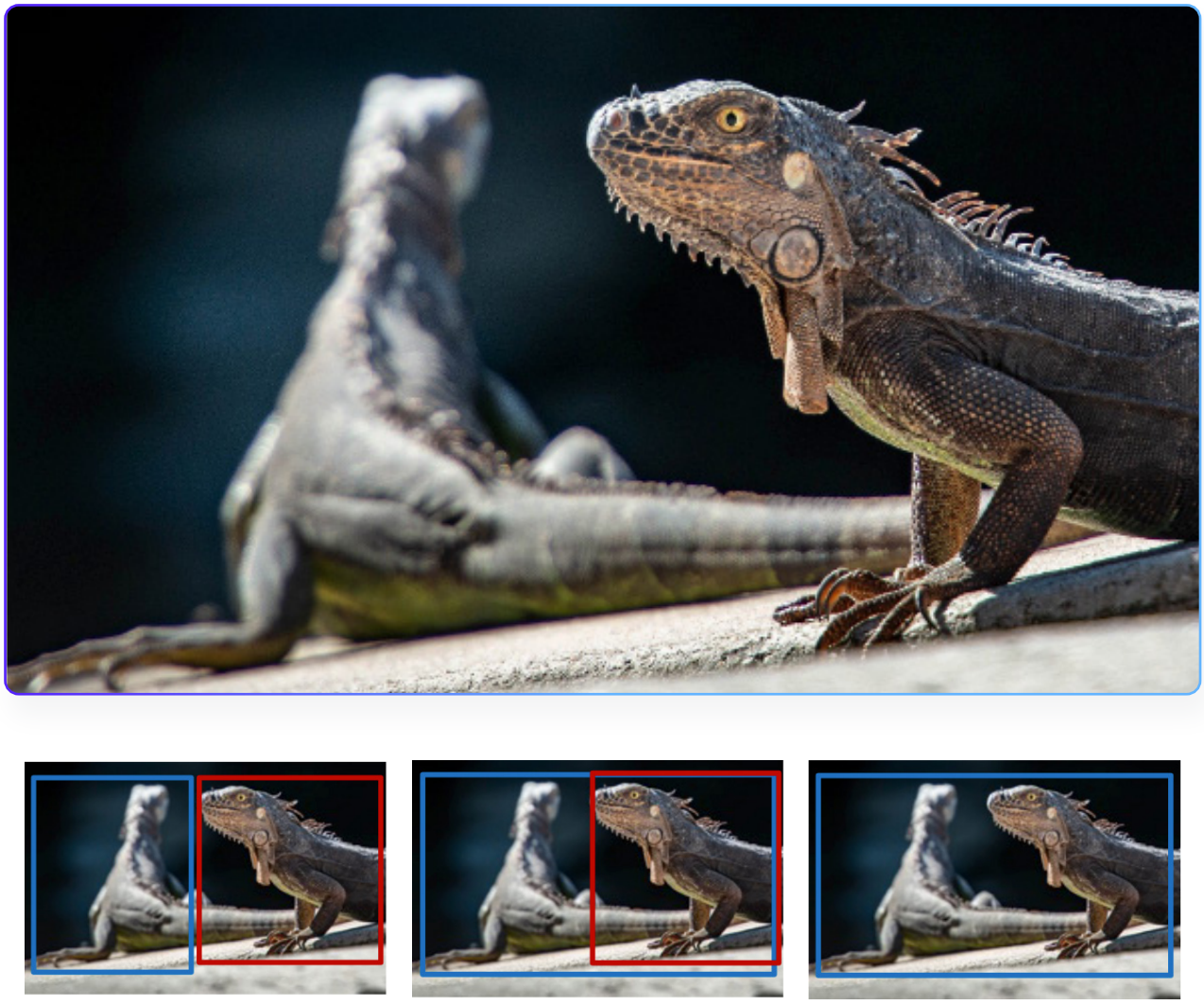


Figure 1. Given the labeling instruction to "use bounding boxes to indicate the position of iguanas", human labelers can output inconsistent labels [Source]

Synthetic Data Will Drive Data-Centric Computer Vision

The advent of synthetic data addresses the issues of manually labeled data. Instead of spending an exorbitant amount of resources and cost collecting and labeling low-quality images, practitioners can turn to synthetic data.

With synthetic data, practitioners can reap a multitude of benefits:

Control

Practitioners have a high degree of autonomy over the content of the images. In the context of a synthetic face dataset, one can customize not only the facial features, but also poses, clothes, and even backgrounds of the face images.

Privacy

Real-world datasets involving personally identifiable information (PII) are laced with privacy concerns. Synthetic data provides the semblance of real-world data without giving out any PII.

Consistency

Since synthetic data labels are generated automatically, they are much more consistent than human labels.

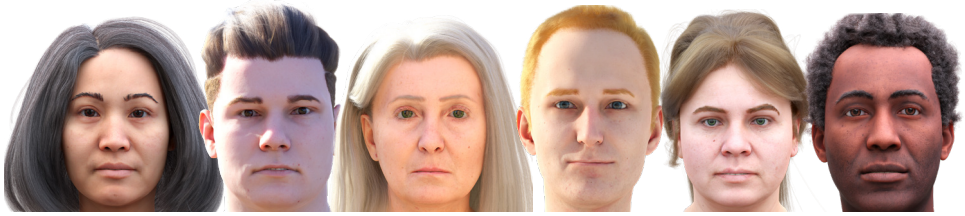
Scalability

Synthetic data can be generated at scale quickly.

Step 1

Generate Synthetic Data

Use the Datagen platform to create synthetic data in a variety of domains: faces, in-cabin automotive, smart office, smart fitness, home security and AR/VR Metaverse. It's easy, all you have to do is design your dataset, click generate and download. Datagen's synthetic data is photo-realistic with perfectly annotated ground truth. If you want to generate synthetic data, you can access our platform for free here.



Images generated by Datagen platform

Step 2

Take an Iterative Approach to Computer Vision with Synthetic Data

Once we have generated our first batch of synthetic data and built a baseline model, it is time to iterate on the model using the data-centric approach (Figure 2).

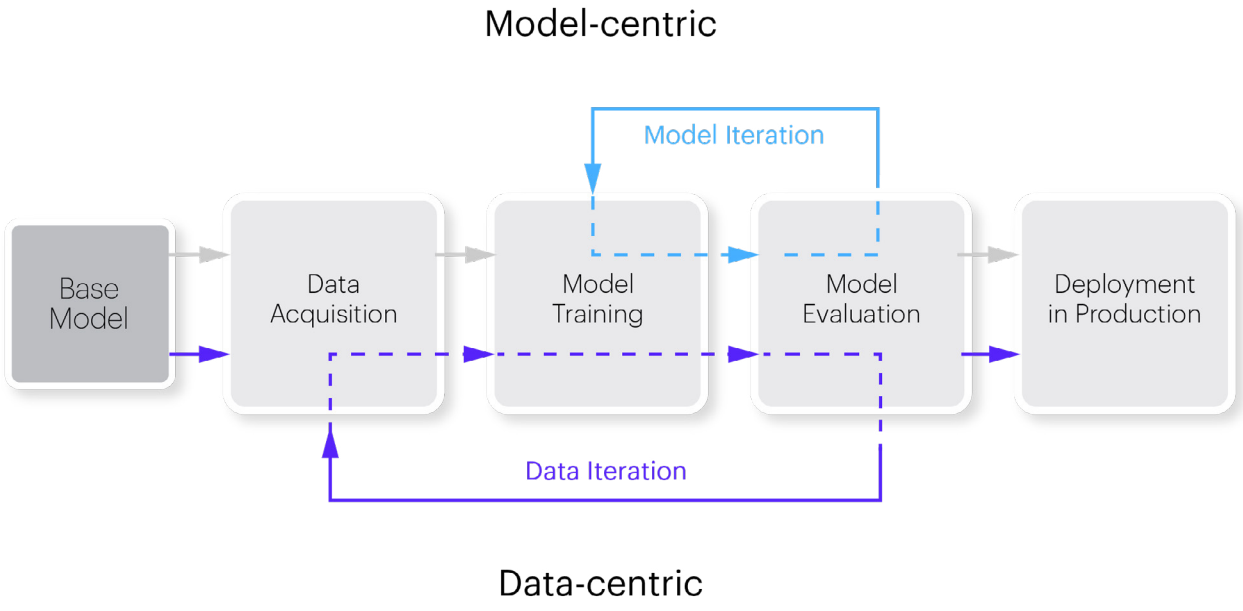


Figure 2. The data-centric approach

The key step here is error analysis or gap analysis. The goal is to identify the gap between the training data and the real-world vision domain. After the baseline model is used to make predictions on real-world test cases, the incorrect predictions are singled out for further analysis.

We can then identify similarities between such failure cases to understand and address the cause of failure. The expectation is that the overall model performance will improve as the model learns to handle the failure cases from the next iteration of data.

Cases to Look Out For in Error Analysis

Practitioners should look out for these common failure cases in the error analysis.

- False positives on a certain class
- False negatives on noisy images
- Misclassification of a certain class
- Failure to locate a certain class

For example, an object detection model of an AV might be unable to detect the road signs in the following scenario. (Figure 3)

Figure 3. Obfuscated road signs [Source]

Step 3

Use Test-Driven Training Approaches

Test-driven development (TDD) is a concept from software development that is gaining momentum in the machine learning community. TDD relies on software requirements being converted to test cases even before the software is fully developed. The software is repeatedly tested against the test cases in its development phase.

Simply put, it involves creating a failing test, writing the code that passes the test, and then refactoring the old code.

When TDD principles are applied to training ML algorithms, we are performing "test-driven training". The intuition here is to create test cases for the AI, train the model on the data, then iterate on the model and/or data until it passes the test cases.

Better yet, the culture of iterative testing should ideally be embedded as part of the modeling process. As practitioners work on improving the model performance, they might also discover more test cases that they would like their models to perform well on. Such additions to the test suite can iteratively boost the model's robustness in the long run.

One of the best practices of test-driven training is to explicitly define the purpose and name of each test case. Each test case should also have a minimum performance threshold as a passing requirement. That way, if an iteration of a computer vision model falls below the passing threshold of a test, practitioners can quickly identify a solution to the problem.

In the book "Thoughtful Machine Learning: A Test-Driven Approach", author Matthew Kirk offered a few concrete tests for test-driven development when training neural networks:

i Seam test

Seams are points of integration between parts of a codebase. We can perform seam tests on a machine learning model by unit testing the data inputs and outputs to ensure that they are valid and within expectations.

For example, if we expect the output of a multi-class prediction neural network to be an array of probability, then the elements must each be in the range between 0 and 1, and all the elements must sum up to 1.

ii Test for shifts in precision and recall

In cases with data imbalances, it is not enough to optimize accuracy because it is likely to result in poor precision and recall. In general, while practitioners focus their attention on optimizing one specific metric (like accuracy), they should also have tests on other metrics (like precision and recall) to better monitor the proportion of false positives or false negatives of the minority class.

Apart from tests suggested by Kirk, we can also create tests to ensure that the model performs well in specific cases and minority classes.

iii Test for specific examples

In the book "Building Machine Learning Powered Applications", the author Ameison called for practitioners to test predictions for specific inputs. This helps proactively detect regressions in prediction quality and guarantees that any model always produces the expected inputs on these example inputs.

For example, a test could be used to specifically identify bounding boxes of deer out of a dataset entitled "500 deers in the headlights at night" with a mean average precision of at least 95%. Failing on this test, alongside other tests in night-time settings, might be symptomatic of a training dataset that is lacking on night-time data.

iv Test for biases against certain subpopulations

In the context of facial recognition models, these subpopulations could be defined by demographics like gender, age, or ethnicity. In the context of an object recognition model, subpopulations could be defined by colors, shapes, or size. Regardless of the model, we strongly recommend practitioners to test for model fairness in their unit test. This prevents the model from potentially discriminating against minority classes from the get-go.

Once such tests are written, we are ready to train our model with synthetic data.

Step 4

Using Synthetic Data as Training Data – Closing the Domain Gap

While synthetic data imitate real-world data closely, it is inevitable for subtle differences to arise. Such a difference in distributions between the real and synthetic datasets is known as the domain gap. Only when such domain gaps are sufficiently small, can we confidently use synthetic data as training data.

Closing the Domain Gap with Domain Randomization

One of the most accessible methods to close the domain gap is to perform domain randomization in the process of generating synthetic data. This approach advertently abandons photorealism and adds non-photorealistic randomness to the photos (Figure 4). This forces the network to learn essential features of the image.

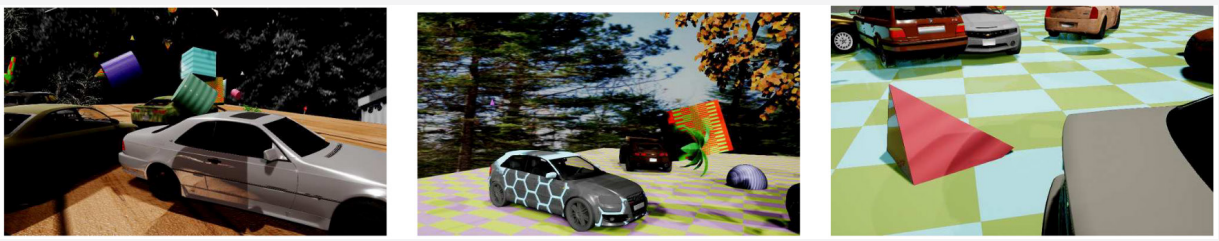


Figure 4: Image of cars generated with domain randomization

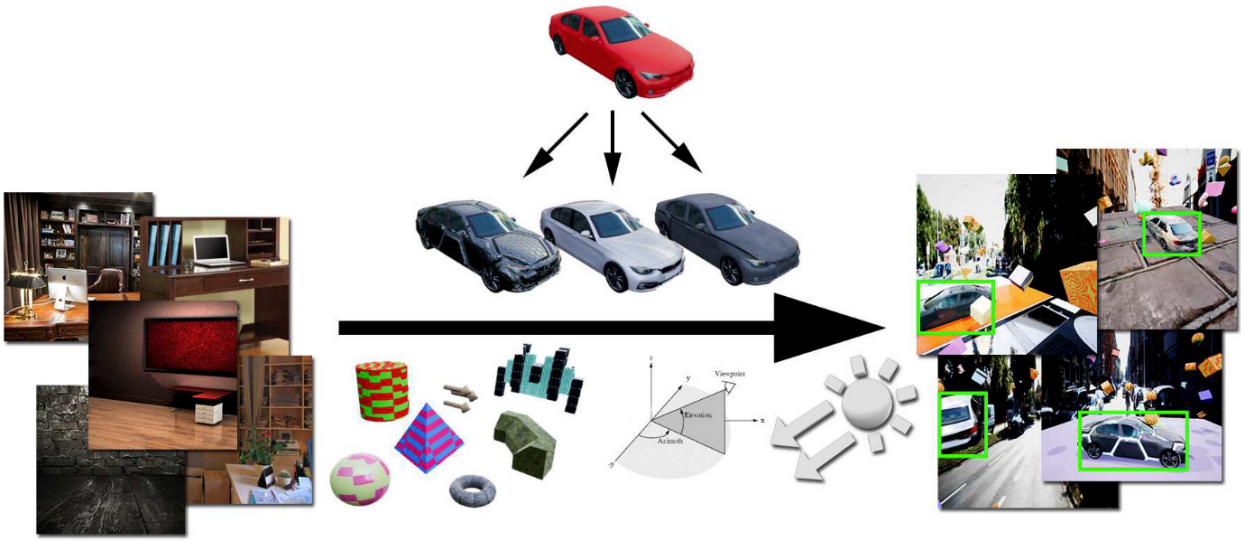


Figure 5: Tremblay et. al. rendered cars (top-center) on top of random backgrounds (left) along with random flying distractors (geometric shapes) in scenes with random lighting and random

Tremblay et. al. illustrated the process of domain randomization as follows (Figure 5)

1. Begin with 3D modes of the object of interest.
2. Add a random number of objects at random positions and orientations.
3. Add flying distractors, which are random geometric shapes to the scene.
4. Add random lights of different types at random positions.
5. Render the scene from random camera viewpoints.
6. Compose the result over a random background image.

Though the images generated with domain randomization are almost cartoonish, the authors demonstrated their effectiveness in bridging the domain gap (Figure 6). The authors have also shown that a model trained on real-world data fine-tuned with domain-randomized synthetic data outperforms that trained on real-world data alone.

Figure 6: A car detection model trained with synthetic images alone (right column) with domain randomization is comparable with that trained with VKITTI (middle column), a real-world dataset.

Given the success of Tremblay et. al., practitioners might want to consider including domain randomization as part of their iterative modeling process. With domain randomization, the model learns to not take into account the randomized visual cues regardless of whether they are useful. For example, a model trained on randomized-colored cars will learn that color is not a useful signal as to whether an object is a car or not. As the model will not be confused by the color of the cars, the network's convergence has improved.

That said, practitioners need to be aware of the trade-off of domain randomization. Since the model has learned to disregard the randomized trait, its performance might be limited in cases where this trait is in fact a useful signal. One should also note that a large amount of data is required to perform domain randomization successfully.

Closing the Label Domain Gap with Label Adaptation

The domain gap could also arise from differences in the methodology used when labels are generated or annotated. Real-world datasets annotated by humans often have imprecise labels (Figure 7), while synthetic data often have precise and deterministic labels. Also, the majority of real-world datasets are annotated with 2D view landmarks, while synthetic datasets are most likely 3D view landmarks (Figure 8).

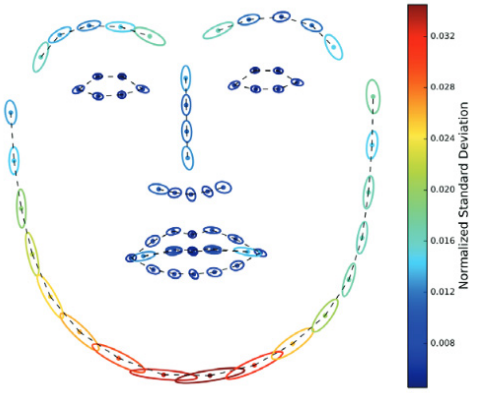


Figure 7. Ambiguity in the way a face is labeled is apparent in the high standard deviation of labels

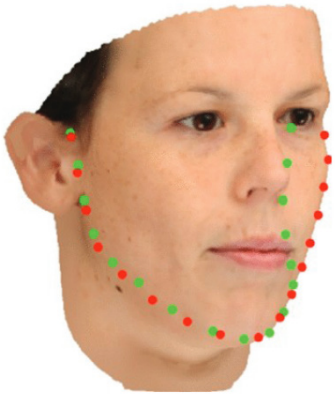
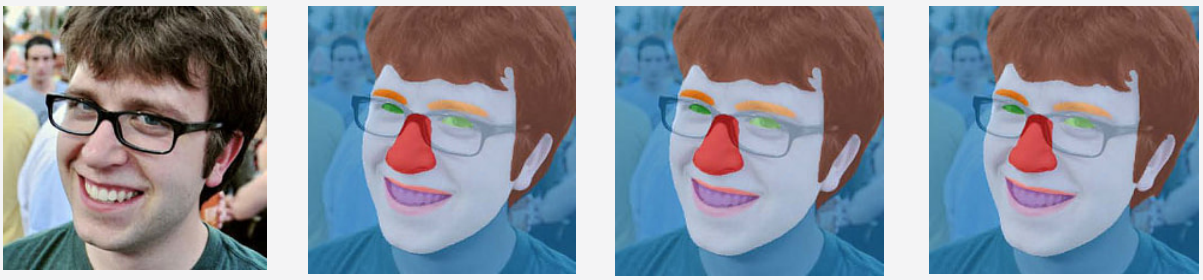


Figure 8. Landmark annotation on face contour differs between 2D (red annotation) and 3D views (green annotation)

Such label domain gaps cause the models trained on synthetic data to output labels that are slightly different from the ground truth. If left unchecked, this domain gap unfairly penalizes models trained only on synthetic data.

To mitigate such label domain gaps, practitioners could use the model label adaptation as proposed by Microsoft's Fake It Till You Make It. In this paper, Microsoft trained a model to transform the labels predicted by the synthetic data model into labels that are closer to the distribution in the real-world dataset. (Figure 9)



(A) (B) (C) (D)

Figure 9. Label output by synthetic data model after model adaptation (C) is closer to ground truth (D) than that before (B)

Step 5

Using Synthetic Data as Test Data to Uncover Biases

Once your model has achieved satisfactory results on the real-world test set, it is highly recommended for you to use synthetic data to enhance your test set.

This is because real-world test sets are often limited. Even a dataset as prominent as ImageNet has "systematic annotation issues" and contains inherent biases against minority groups.

When test sets are made only out of real-world data, they could inherit such biases too. A real-world test set that does not include any edge cases or instances of minority classes will not be able to sieve out biased models.

If such flawed models make it to production, they become ticking time bombs waiting to cause harm to vulnerable communities. That is why even Tesla uses synthetically generated data to create edge cases and test their models against it (Figure 10).

Figure 10. Tesla generates scenarios to test its AV [Source]

We recently published the paper Using Synthetic Images to Uncover Population Biases in Facial Landmarks Detection at the NeurIPS Data-Centric AI Workshop held in 2021. In this paper, we demonstrated the use of synthetic data to detect biases. Here, we tested a facial detection model trained on real-world data against one trained on a combination of real-world and synthetic data. More concretely, here are the steps:

1. Use the DLIB package to detect facial landmarks.
2. Use real-world datasets like CelebA as the test sets.
3. Compute the normalized mean error (NME) of the model on each face.
4. Stratify faces based on physical attributes, like men vs women, young vs old, and white-skinned vs black-skinned. Calculate the mean NME on each group, and identify statistically significant biases. If the model's mean NME is significantly higher for women than for men in the CelebA dataset, then we say that the model is biased against women.
5. Repeat steps 2 to 4 using Datagen's synthetic datasets.
6. Evaluate whether Datagen's synthetic data and CelebA revealed the bias of the model.

This methodology revealed that the model is biased against minority classes like women, the elderly, and black-skinned individuals. In particular, the bias is more apparent when the algorithm is tested on Datagen's synthetic data as compared to CelebA.

As AI fairness becomes an increasingly pressing issue, practitioners might want to enhance their test sets with synthetic data to detect the model's weaknesses.

Step 6

Mixing Real and Synthetic Data in Training and Test Sets

So far, we saw how a train set with only synthetic data is used in the modeling process. Can we combine hand-labeled data with synthetic data to create a high-quality training dataset?

At Datagen, we confirmed such a hypothesis through a series of experiments on facial landmark detection models. We found that a training set consisting of varying amounts of real and synthetic data can successfully produce models that are just as performant as models trained only on real data.

In the paper Facial Landmarks Localization Using Synthetic Data, we explored two strategies for combining the real and synthetic data.

- In the 'mixing' strategy, we trained the model with a combination of label-adapted synthetic data and real data
- In the 'fine-tuning' strategy, we trained the model on synthetic data, and then tuned the model with real data

The performance of these two strategies under different proportions of real and synthetic data is shown in Figure 11 below.

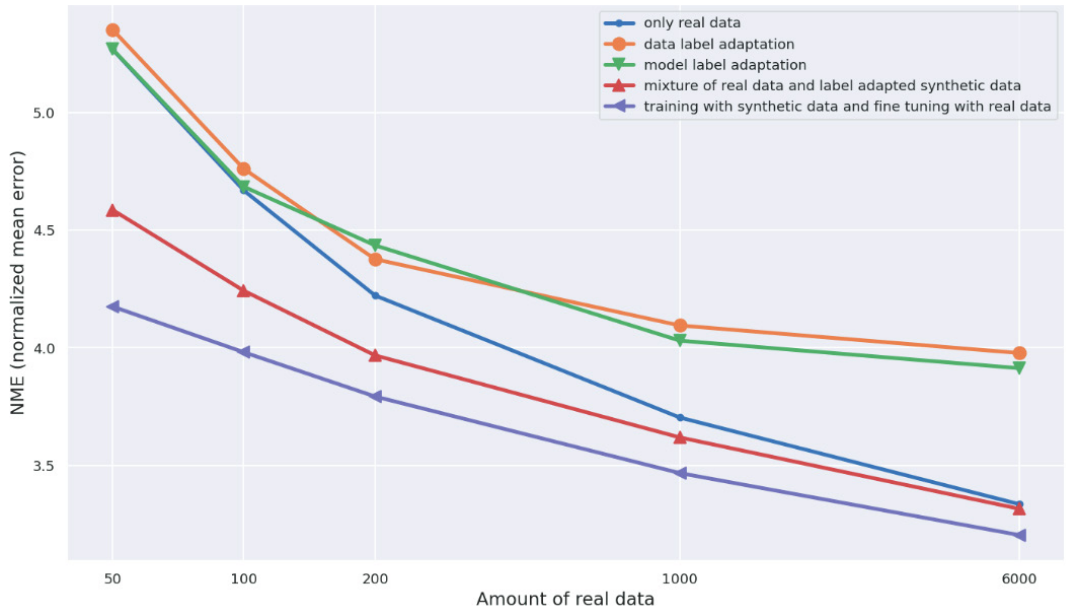


Figure 11. Performance of landmark detection models trained with different strategies

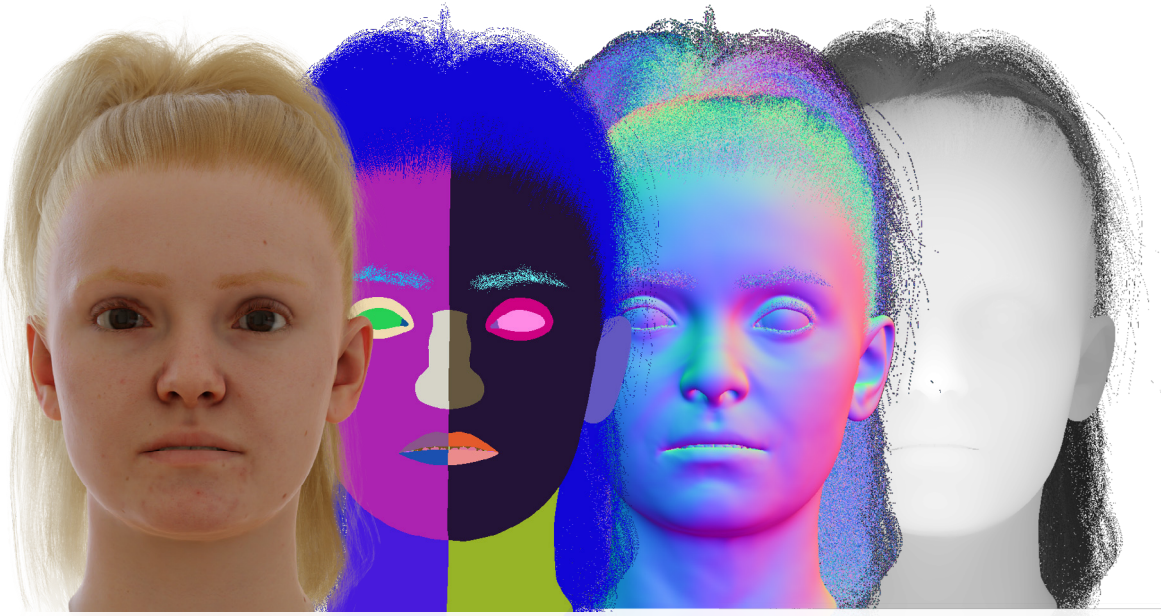
In general, we made the following observations:

- Both strategies perform better than the model trained on real data only, illustrating the value of combining synthetic data with real data.

- The fine-tuning strategy outperforms the mixing strategy. Thus, we recommend practitioners use the fine-tuning strategy when combining real and synthetic datasets.

- The performance of the fine-tuning strategy with 50 real data points is equivalent to that of the real-data-only strategy with 250 real data points. In short, fine-tuning reduces the amount of real data required significantly.

This 6-step guide is only a starting point in your synthetic data journey. Get started with a free trial of the Datagen platform today.



Images generated by Datagen platform

Datagen Synthetic Data Solution

Datagen is the Data-as-code company, a category-defining solution for the Computer Vision synthetic data market. Datagen's approach turns the heavy operational process of visual data collection and annotation into an easy-to-control programmable user interface, enabling Computer Vision teams to generate data and train and evaluate models, across the development lifecycle. Fortune 500 companies rely on Datagen's self-service human-centric synthetic data platform and API to develop their future products in the worlds of AR/VR/Metaverse, In-cabin Vehicle Safety, IoT Security and more. Founded in 2018, Datagen is led and backed by world-renowned AI experts.

Start Your Free Trial