A Process to Enhance Deep Machine Learning Models with Custom Training Data

Running Head: Process to Enhance Deep ML Models

Beta Build Draft 0.0.0.5

Julian Klein
11/10/20

Julian Klein
11-10-20

A Process to Enhance Deep Machine Learning Models with Custom Training Data

The following guide describes a process that will generate highly customized data for deep machine learning models. It is an efficient way to render a very large set of training data images in a very short amount of time, while incurring little or no cost, thanks to open source software.

Conceptualization

This white paper aims to keep the language simple and easy enough for a layman to understand, but it is imperative that we share the same language and definitions. In the subsequent pages, the term artificial intelligence (AI) describes the general goal of using computer algorithm to predict or recognize novel objects. The way to develop or train such an algorithm is referred to as Machine Learning (ML). If the process of training the machine happens on multiple levels or planes, it becomes Deep Machine Learning (DML), thanks to mathematical tensors. The definition of a tensor is actually outside the scope of this paper, but just imagine that it allows an algorithm to find similarities on multiple planes, like the shape and also color of an object, by comparing and extrapolating information from a large set of training data, for example, thousands of images. TensorFlow is “a flexible data flow-based programming model” (Google, 2019, p. 16). The described process also involves Blender 3D, which is a “free and open source 3D creation suite” and “supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation" (Blender, 2021, p. 3). The software tools and the described process are solely focused on object or image recognition and rendering data that facilitate deep machine learning.

The Use-Case Intro

In 2019, curiosity got the best of me, and I decided to learn what AI is actually about. I already had a use-case, too. My friend had recently started a pistachio farm in the Mojave desert, which sounded like a terrible idea, so naturally I helped him plant his trees. What I learned was that grafted pistachio trees have rootstock, that sprout rootstock branches. These rootstock branches suck valuable nutrients and water from the desirable branches of the grafted stock above. The problem is currently solved with manual labor, by trimming the rootstock branches, also known as suckers. That year, I did a lot of trimming and a little thinking. In my dreamy little mind, a small farm robot could use a camera, drive around, and do the work for us... if it only knew how to recognize those pesky suckers. It didn't take long and I started thinking about AI image recognition for all sorts of objects. Plants, with non-uniform branches and leafs, were tricky compared to solid-shaped objects, like weapons. The latter had utility for machine learning too. After all, instead of looking for people, why not look for threats? We've learned in recent years that the potential for racial bias is damning for the algorithm behind it; and rightfully so. Imagine if we could teach image recognition software to find full or partial matches for handguns or rifles. Ultimately, I decided to focus on the difficult objects, the plants, and on a new data model training process as the solution. My research question became: how can we generate better ML training data with very limited resources?

About the Process

Years before helping my friend start his farm, I had launched a business that created 3D models, animations, and illustrations, for purposes like surgical, medical training. That is how I came to work with Blender 3D on a Kubuntu Linux system; which was awesome. Blender software is used for 3D anything, so it does a lot of things. I would call it the most convoluted and impressive software tool I have ever worked with. Many users don’t even realize that it can also render images from the command-line interface (CLI) and that it can be customized with python programming language. Subsequently, users can write a script that renders thousands of images of a given 3D model, while constantly changing variables like, camera angle, lighting conditions, colors, or even the shape and contours of an object. In machine learning these variables would likely be referred to as parameters, not to be confused with the hyperparameters that follow later in the process pipeline. If I create a simple 3D object, like a rootstock branch and leaf, I could practically auto-generate a wealth of training data for machine learning, literally over night. The 3D objects could be rendered against various backgrounds to match their use-case environment, like a desert landscape High Dynamic Range Image (HDRI). The rendered output could range from abstract to photo-realistic. The best part, the process is so affordable and simple, that even I could do it.

Custom Parameters for Training Data Production

Software engineers and architects generally start big projects by looking at the requirements that might solve a given problem. In our use-case, we know that the product will be a robot, and it would be using a Pi camera with 2MP, because that’s what I had lying around. Blender allows users to mimic the camera settings of a Pi camera, by specifying custom camera settings within a given .blend file. For example, a user can specify that the software's camera should use an active array (resolution) of 1600 by 1200 pixels, a ¼" lens, and position the camera 30 centimeters above the ground, and/or the line of view perpendicular to the ground. This customized output imitates the view of a robot's camera and hopefully you're beginning to see how customization can yield better image recognition training data; but we're only getting started.

The customization requirements for the object would naturally be things like, lighting and surface conditions, including color, reflectivity, roughness, and so on. To get a glimpse of the wealth of available variables, look at the Principled BSDF Node within the node editor of Blender 3D. BSDF stands for bidirectional scattering distribution function. Features like BSDF are key to physics based rendering (PBR), meaning the software's render engine actually calculates rays from the light source, hitting a surface, being absorbed, scattered, or reflected, and finally captured by a virtualized camera lens. The output Blender's cycle render engine creates is amazing and mind-boggling.

The software used in this process are all open source. TensorFlow and Keras are licensed under Apache 2.0. OpenCV used to be licensed under BSD 3-clause and is now also Apache 2.0. Blender 3D is licensed under GNU GPLv3, but the output is not covered by the GPL license (Blender, 2022), so the creative work and output belongs to the user.

Pre-Mature Conclusion

This method of producing custom data for deep learning lends itself to more than agri- or horticultural applications. The key finding is that Blender 3D can produce life-like renderings that mimic reality and produce thousands of highly customized images that help train better machine learning models.

Proof of Concept

The following process is guided through an in-depth explanation and screenshots that depict the entire procedure from creating a 3D model, to rendering images, to training an AI model, and testing the ML results in the field.

Step 1: Create a 3D model of a branch and leaf

Step 2: Automate the change of parameters or variables (via keyframes or python)

Step 3: Render the output images and split them into sets (write a bash script for this)

Step 4: Write an ML python script that imports TensorFlow, imports the data set, explores the data, preprocesses the data, builds, compiles, and trains the model, then evaluates accuracy, makes predictions, and then plots images, as well as their predicted and true labels.

Step 5: Bring a laptop to the farm and see if it can tell a pistachio rootstock from the grafted stock.

About this Author and License

I'm a generalist and not particularly deep in my AI knowledge, but maybe someone might find this process useful. In any case, if this idea is sound and novel, it is probably too important to hog, own, or reserve; plus it leans on truly great ideas that came before mine. In the spirit of the underlying software I used (Linux, TensorFlow, Blender, etc.), I would like to license this process under the open-source Apache License 2.0.

Disclaimer: This is an early draft, incomplete, missing paragraphs, and in desperate need of proper citations. If you like what you've read, please encourage me to finish the paper.

References

Blender Foundation https://docs.blender.org/api/current/info_overview.html (Retrieved 2022)

https://www.blender.org/support/faq (Retrieved 2022)

https://www.blender.org/about/ (Retrieved Nov. 2021)

Google https://www.tensorflow.org/