



Generating AI “Art” with VQGAN+CLIP

Created by Phillip Burgess



<https://learn.adafruit.com/generating-ai-art-with-vqgan-clip>

Last updated on 2021-11-15 08:26:34 PM EST

Table of Contents

Overview	3
Basic Use	4
• Starting and Stopping Jobs	5
• First Time Through	6
• Uploading Files	6
Piloting the Weird	7
• “Selección de modelos a descargar” (“Selection of models to download”)	7
• “Parámetros” (“Parameters”)	8
Execute!	10
• Hacer la ejecución... (Do the execution ...)	10
• Genera un vídeo con los resultados (Generate a video with the results)	12
Troubleshooting and Notes	13

Overview



Reading social media and science articles as of late, you’ve probably had the misfortune of encountering surreal, sometimes nightmarish images with the description “VQGAN+CLIP” attached. Familiar glimpses of reality, but broken somehow.

My layperson understanding struggles to define what VQGAN+CLIP even means (an acronym salad of Vector Quantized Generative Adversarial Network and Contrastive Language–Image Pre-training), but [Phil Torrone deftly describes it as “a bunch of Python that can take words and make pictures based on trained data sets.”](https://adafruit.it/TFc) (<https://adafruit.it/TFc>) If you recall the Google DeepDream images a few years back — where everything was turned into dog faces — it’s an evolution of similar concepts.

GANs (Generative Adversarial Networks) are systems where two neural networks are pitted against one another: a generator which synthesizes images or data, and a discriminator which scores how plausible the results are. The system feeds back on itself to incrementally improve its score.

A lot of coverage has been on the unsettling and dystopian applications of GANs — deepfake videos, nonexistent but believable faces, poorly trained datasets that inadvertently encode racism — but they also have benign uses: upscaling low-resolution imagery, stylizing photographs, and repairing damaged artworks (even speculating on entire lost sections in masterpieces).

CLIP (Contrastive Language–Image Pre-training) is a companion third neural network which finds images based on natural language descriptions, which are what’s initially fed into the VQGAN.

It’s heady, technical stuff, but good work has been done in making this accessible to the masses, that we might better understand the implications: sometimes disquieting, but the future need not be all torches and pitchforks.

There's no software to install — you can experiment with VQGAN+CLIP in your web browser with forms hosted on [Google Colaboratory \(https://adafru.it/TXb\)](https://adafru.it/TXb) (“Colab” for short), which allows anyone to write, share and run Python code from the browser. You do need a free Google account, but that's it.



Basic Use

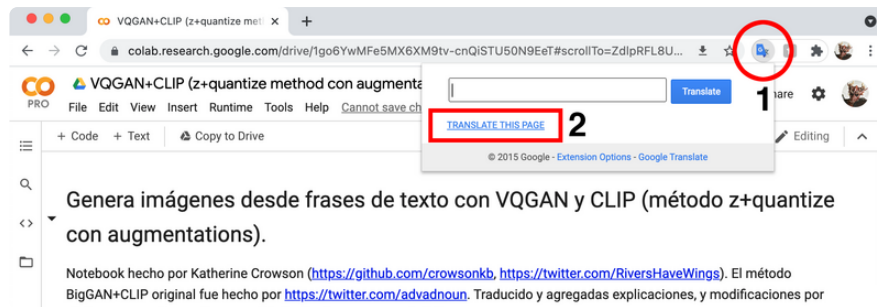
Here is a link to Katherine Crowson's (<https://adafru.it/TVe>) [project on Google Colab \(https://adafru.it/TVe\)](https://adafru.it/TVe) (opens in new window). Access is free to anyone; you do not need a Colab Pro account to try this out (just a normal free Google account), though resources are more limited to free users. I could generate 3–4 short clips per 24 hour period before it complains.

Google Chrome is recommended as it's known to be fully compatible. Safari (perhaps others) can't download the MP4 videos produced in the final step.

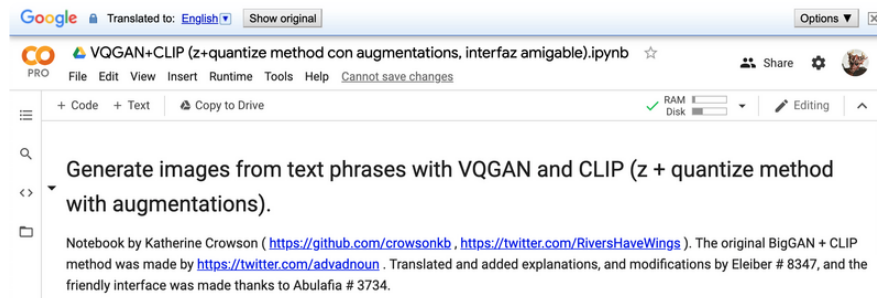
Before jumping in, best to familiarize yourself with some basics of the Colab forms' interface...



Crowson's form is in Spanish, while the code and text output is partly English. One can get by on lexical similarities and a lot of this being jargon anyway...but if you'd prefer, Chrome has a translation feature. Click the icon just to the right of the URL box, then “TRANSLATE THIS PAGE” to activate it.

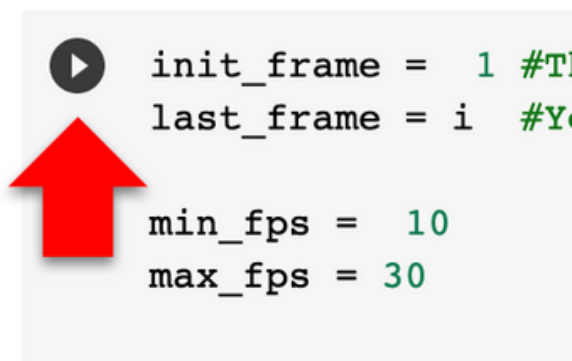


Translated...

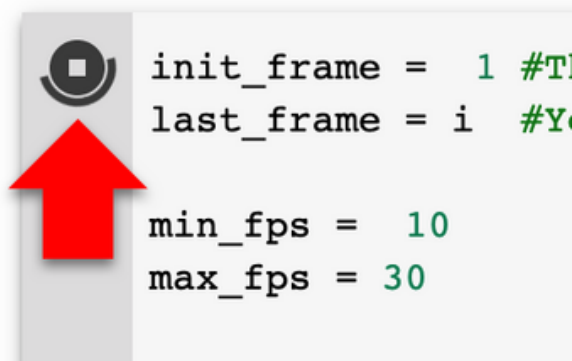


Starting and Stopping Jobs

If you want to generate a video



If you want to generate a video

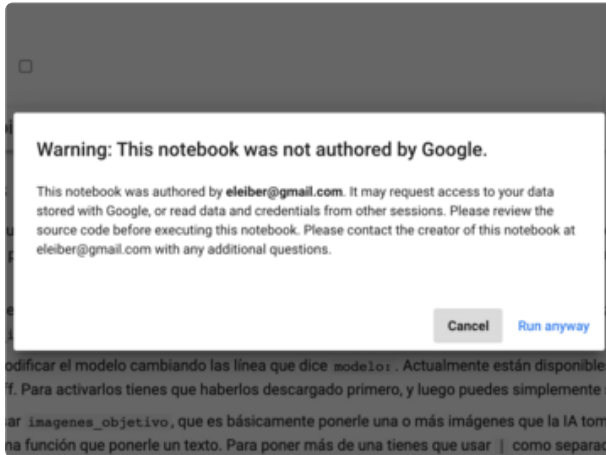


There is a sequence of steps, which will be run top-to-bottom. Each step has this “run” button, which changes to a spinning “busy” indicator while running — clicking that during a run cancels the corresponding process.

Use these slowly and deliberately, do not “mash” the buttons. Some processes are slow to respond, and excessive clicking will cancel and then restart the process, losing interim data you might have wanted to keep! Also, a first click will sometimes just scroll that item to the top of the window and not take any action. Click, think, click again only if required.

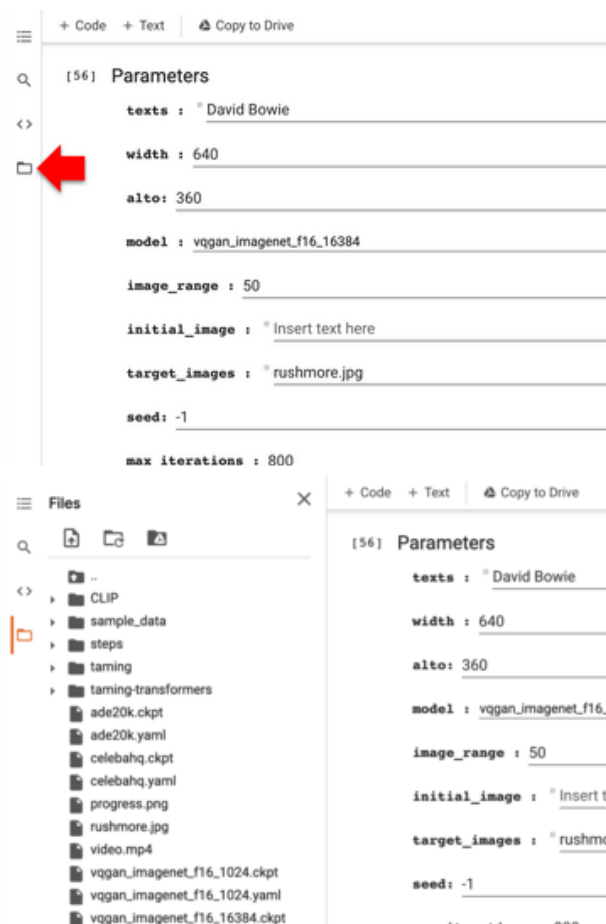
First Time Through

But...rather than running each step manually, I find it easier to set up parameters first (explained on next page) and then use Colab's "Run all," which powers through all the steps in sequence. You'll find this at the top in the Runtime menu. On subsequent trials, you can then re-run the individual pieces as needed.



The first time running any step (or "Run all") you'll get this warning box. That's normal and it can be dismissed with the "Run anyway" button. The project's been tested by a great many at this point, and the software is running "sandboxed" on Google's servers, not your own system.

Uploading Files



Certain VQGAN parameters can accept image files as input. To transfer files from your machine to Colab, click the folder icon in the left margin, which unfolds into a file selector, then drag and drop your image files into this list. Click the icon again to collapse this section.

Any files you transfer there are not permanently stored. Closing the browser window will end the session and remove anything in the sandbox; you'll start from a clean slate on your next visit.

Piloting the Weird

As mentioned on the prior page, a series of jobs are run top-to-bottom, but on the first pass we'll "Run all" to automate this. Some of the initial jobs are non-interactive, so scroll down a bit and we'll start fiddling mid-form before setting it in action...

"Selección de modelos a descargar" ("Selection of models to download")

This is where one selects one or more pre-trained models for the VQGAN. These models were assembled by various research groups, trained from different sources, some for broad use or others tuned to specific purposes such as faces.



Only one model is active at a time, but you can download more than one if trying some A/B comparisons through multiple runs. Some of these models are truly massive or are hosted on bandwidth-constrained systems, so choose one or two carefully, don't just download the lot.

By default, imagenet_16384 is selected — it's a good general-purpose starting point, trained from a large number of images prioritized by the most common nouns.

You can Google around for explanations on most of these, but for example...

ade20k is tuned to scenes, places and environments. This might be best for indoor scenes, cityscapes or landscapes.

ffhq is trained from a set of high-resolution face images from Flickr. You may have seen this used to make faces of "nonexistent people."

celebahq is similar, though specifically built from celebrity faces.

Whatever model(s) you select here, you'll specifically need to activate one of them in a later step...

“Parámetros” (“Parameters”)

The VQGAN model does all the “thinking,” but this is where you steer the output. If doing multiple runs, you’ll be returning to this section, editing one or more values, and clicking the “run” button to validate the inputs (but not yet generate any graphics).

The fields in this form include:

textos (texts): use this field to describe what you’d like to see in plain English. The “CLIP” part of VQGAN+CLIP processes text into images to feed the “VQGAN” part.

More detailed is generally better. “Carl Sagan” could go anywhere, but “Carl Sagan on a beach at sunset” provides a lot more context to work against.

With the generalized models, a popular addition is “in the style of,” where you can transmogrify a subject into a facsimile of some artist’s work. “Robot pterodactyl in style of Dali.” “Heckin' chonker cat in style of Leyendecker.” “Monster Squad movie in style of Lisa Frank.” Amazing how many styles it’s able to emulate. Not just individuals, try “Unreal engine,” “stained glass,” “Don Bluth,” “Pixar” and others. The more absurd the combination, the better.

You can also separate prompts with a vertical bar. “McRib | low poly” or “San Diego Supercomputer Center | HR Giger”

Anything with “Adafruit” eventually sprouts LEDs and pink hair.



ancho, alto (width, height): dimensions of the resulting image or video, in pixels. The default values are both 480, for a modest sized square image. Larger images take

geometrically more time and resources to process, so consider staying close to this pixel count. For example, I use 640x360 a lot ... it's the same number of pixels, but a 16:9 aspect ratio, great for video or for Twitter's single-image crop.

`modelo` (model): which VQGAN model to use for image reconstruction. Different models are tuned to different purposes. The corresponding dataset must have been previously downloaded (see "Selection of models" above).

`intervalo_imagenes` (image_range): how many iterations of the VQGAN between preview images displayed in the browser, letting you see progress as the subject comes into focus. Every iteration is actually stored for producing a video later, this is just how often we see a frame in progress (you can also right-click and save any of these individually, if you just want a few stills instead of video). Default is 50.

`imagen_inicial` (initial_image): normally this field is blank and the process starts with a little random noise, and the VQGAN interprets this like you or I finding faces in clouds. Optionally here, rather than noise, you can provide an image as a starting point and have a little say over composition. This is also sometimes used to feed the output of one VQGAN model as the input into another. As explained on the prior page, images can be uploaded in the "Files" section (left margin), and referenced in this field by filename.

`imagenes_objetivo` (target_image): also normally blank, here you can optionally provide an image to steer the VQGAN toward, rather than from.

You can use either or both of these fields, or even the same image for both ... sometimes the process quickly strays from the original composition, but this is a way to help keep it on track, as seen here:



`seed`: this provides a starting point for the random number generator. The default of -1 tells it to use a random seed — you'll get different results each time, even with all other values the same. Supplying a number allows prior results to be reproduced. If you started random, but like the results and want to reproduce it at a different size or make a longer video, you can see the randomly-chosen seed when running the "execute" job (explained on next page), and copy-and-paste that into the parameters for the next run.

max_iteraciones (max_iterations): How many VQGAN iterations to run before stopping. The default of -1 has it run indefinitely (you can stop the process manually when satisfied with the output). I usually set an upper limit here, maybe 800 or 1000 (most images “solidify” well before that), so I can “Run all” and do other things while it works, without the job running rampant with resources.

If you selected “Run all,” and if you set a max_iterations value (or interrupt the process manually), the project will continue to assemble and download a video of all frames up to that point. In subsequent runs, you can do those steps manually.



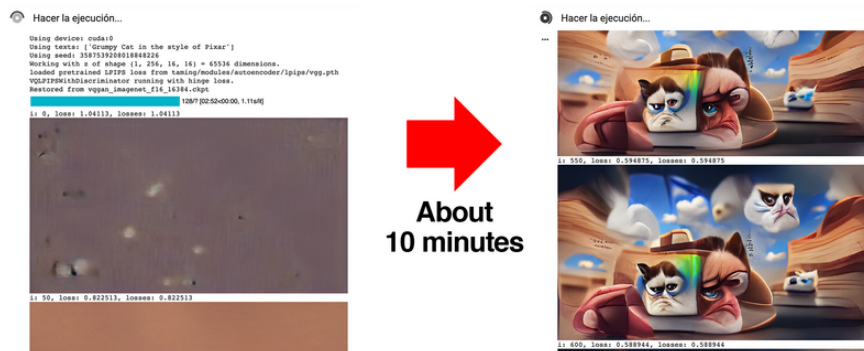
Execute!

Hacer la ejecución... (Do the execution ...)

Double-check all your parameters, then click the Run button on this job (if using “Run all” for the first time, this will happen automatically). Remember if making changes from a prior trial, you should re-run the Parameters job to validate the inputs before running this one.

After a moment for initialization, you'll see a muddy rectangle (if "initial_image" was blank), or your starter image. The VQGAN model will then iteratively refine this mud/ image toward the situation you described (and/or the target_image provided).

Have patience. At the default 480x480 size, it'll take about one second per iteration, and at the default 50 interval you'll get about one visual update per minute. The first couple of updates will be vague nonsense, but with each successive update you'll see the scene coming into focus. If you don't like where it's headed, stop the job, edit any parameters, then run the Parameters job and the Execute job to begin again.



There's a certain point where a scene will "congeal" and further iterations make little difference. This will vary among the different models, parameters and just random chance...but very occasionally it will surprise you with a sudden shift in overall color or texture. With the imagenet_16384 model, I typically set a limit of 800 iterations, but the scene is usually established about halfway through and I might stop it earlier.

Accompanying each preview image you'll see a "loss" value which gradually decreases with each successive image. In theory, a value under 0.5 means the discriminator finds the scene "plausible." However, the image-to-image change tends to decelerate and good chance won't cross that threshold. Also, I've only got so many hours in my life and can't afford to wait forever, I just want to see weird distorted cat pictures, y'know?

Another good reason for limiting the size and number of iterations is that session time and system resources are limited, especially in the free mode. You might squeeze our three or four attempts in a 24 hour period, then have to wait for the following day to experiment more.



Genera un vídeo con los resultados (Generate a video with the results)

If you just want a still image (or a few along the way), you can right-click any of the in-progress execution images and “Save as...” This a PNG image file, you’ll need other software if you want to convert to JPEG.

For video, a couple more jobs must be run. If “Run all” was selected, these last steps will happen automatically once the Execute job finishes (iteration limit reached) or is cancelled in-progress.

▼ Genera un vídeo con los resultados


Si quieres generar un video con los frames, solo haz click abajo. Puedes modificar etc.

```
init_frame = 1 #Este es el frame donde el video empezará
last_frame = i #Puedes cambiar i a el número del último frame q
min_fps = 10
max_fps = 30
```

(trimmed for brevity)

```
print("El video está siendo ahora comprimido, espera...")
p.wait()
print("El video está listo")
```

Generando video...

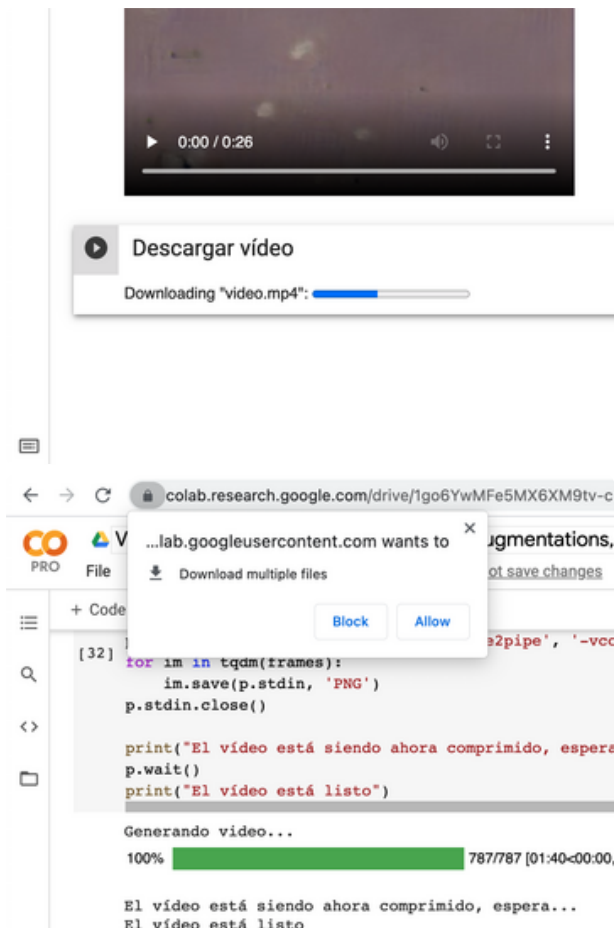
13%  101/787 [00:12<01:29, 7.63it/s]

The next job below “Execute” is “Generate a video...”

When run, this will coalesce every image iteration (not just the preview images, but every frame in-between) into a MP4 video file.

It takes a couple minutes to complete, and there’s a progress bar while it works. Once the conversion is complete, the bar turns green.

Next there’s a “View video in browser” job, but I find it’s generally not worth the effort. The preview images along the way give a pretty good impression whether the results are worth keeping. Instead, skip down to the last job on the page, “Descargar vídeo / Download video.”



Just click the “Run job” button and this will download to your computer, wherever the browser preferences deposit downloaded files.

The first time you run this in a session, you might get a confirmation dialog to allow downloaded files. That’s fine, just the browser being vigilant. It’s easy to miss while distracted with all the fun graphics though. Allow downloads and be on your way.

Keep in mind: if you run another “Execute” job, you’ll need to re-run “Generate a video” prior to downloading, otherwise you’ll download your prior video. The MP4 isn’t produced automatically each time.

Troubleshooting and Notes

ModuleNotFoundError when running celebahq or ade20k models (possibly others)

Some of the VQGAN models rely on Python modules that aren’t installed by default, but it’s a quick fix to set this up...

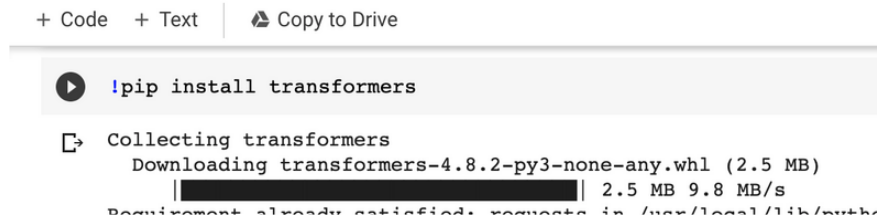
Near the top of the window are some options, “+ Code,” “+ Text” and so forth. Click on “+ Code” to insert a new cell below whatever job currently has focus. Its position in the sequence really doesn’t matter.

In the resulting cell, type:

```
!pip install transformers
```


(Include the initial “!” character, that’s important.)

Then click the “Run job” icon, and you’ll see a progress bar as it installs. Only takes a moment.



The screenshot shows a code editor interface with a toolbar at the top containing "+ Code", "+ Text", and "Copy to Drive". Below the toolbar, a terminal window displays the command `!pip install transformers`. The output shows the process of collecting and downloading transformers, with a progress bar indicating 2.5 MB downloaded at 9.8 MB/s. The text is partially cut off at the bottom.

```
+ Code + Text Copy to Drive
!pip install transformers
Collecting transformers
  Downloading transformers-4.8.2-py3-none-any.whl (2.5 MB)
    |████████████████████████████████████████| 2.5 MB 9.8 MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.7/site-packages
```

That’s it! Now you can go back and run the “Execute” job. You only need to run this “pip” operation one time for the whole session, it sticks around until you close the window.

NameError: name 'torch' is not defined (or sometimes 'argparse')

Occasionally you’ll get this when a session idles out or becomes disconnected. Re-run the “Instalación de bibliotecas” (“Installation of libraries”) and/or “Carga de bibliotecas y definiciones” (“Loading libraries and definitions”) jobs, then try the Execute job again.

If that doesn’t fix it, sometimes you just have to reload the whole page and start over.