Generating novel fruits from text descriptions using Variational Autoencoders

Jeremy Dohmann

May, 2020

1 Introduction

Using deep learning to represent joint distributions over images and text has a long history in computer vision, with large-scale successful attempts at image classification and image captioning dating to the early 2010s [1, 2, 3].

Image classification and image captioning often focus on learning mappings from entity and action classes to their visual analogues. While these are unarguably the most important building blocks of image understanding, I am interested in deep learning's ability to learn the relationship between language and images at a level below the object class. In particular, I am interested in using text descriptions of object classes to learn the relationship between instances of a class and the individual visual features that constitute instances of that class (or in semantic terms, learning the relationship between the intensional definition of a class and its visual instantiations). For example, by learning a mapping from text descriptions to images, we may be able to learn meaningful representations for concepts such as color, shape, and texture.

There exists at least one substantial work on this subject, in which researchers used a custom text embedding and GAN architecture to produce novel birds and flowers from textual descriptions [4]. They used a highly sophisticated architecture which hinged on a discriminator both determining the authenticity of images and determining whether the text and synthetic images were well-suited pairs. Additionally, their study benefited from having a robust, highly curated data set consisting of multiple pairs of hand-written visual descriptions for each image in their set. [1]

2 Problem and approach overview

Similar to [4] I was interested in learning to produce synthetic images from textual descriptions. By contrast with their work, I was much more interested in investigating the relationship between object classes, their visual instantiations, and natural language. Whereas their work focused primarily on one-off pictures of birds with ad-hoc visual descriptions of that particular image, I was interested in learning to map from the definition of an object class (e.g. 'Coconut') to particular visual instantiations of that definition. This understanding of the problem is much more deeply rooted intensional semantics which concerns itself

[1] Code available at https://github.com/jcd2020/text2image

with defining what an object class is by stipulation of a set of criteria that instances of that class must meet: e.g. the essence of a coconut can thought of as ‘tropical fruit with a round, hairy, hard shell’.

To that end, I wanted to design a system, which for some number of object classes, could learn to map textual definitions of those objects to synthetic images of objects appearing to belong to that class.

Rather than working with a GAN architecture, I decided to go with a Variational Autoencoder (per Colin’s suggestion) because they are easier to work with, require less data, make it easier to condition the output based on specific inputs, and are believed by some to be better at producing truly synthetic images. I pulled some of my code from the following introduction to VAEs. VAEs differ from traditional autoencoders in that they parametrize the latent space as a multimodal gaussian distribution, where a particular input maps to a mean vector and standard deviation vector. VAEs are successful at learning meaningful latent spaces because they incorporate a loss term which enforces continuity across the space. This means that separate object classes don’t get mapped to wildly different parts of the latent space, resulting in vast areas of dead space corresponding to no class in particular. This issue of ‘discretizing’ the space is particularly bad in an application such as mine where one already runs a high risk of overfitting due to the constraints on the amount of textual data available.

Though the textual descriptions in [4] are of very high quality and hand-tailored for the task, I decided to forego their data set in favor of the Fruits 360 data set, because the images are extremely high quality and it boasts a large number of discrete classes. Additionally, there are relatively few axes along which fruits vary in appearance, and fruits are much more structurally simple than birds and flowers [5].

In order to source textual descriptions, I pulled descriptions from specialtyproduce.com, a website with hundreds of in-depth descriptions of various fruits and their cultivars. Their descriptions are not strictly visual, and thus contain a great deal of spurious information - but they are also a good representation of open source texts that one might encounter in the wild.

For instance, the text description for Kiwi is below:

> About the size of an egg, it is wrapped in a russet-brown thin skin with short rather stiff hairs. The kaleidoscope-like almost glistening emerald green firm pulp is dotted with a large amount of dark nearly black tiny edible seeds that create this fruit’s characteristic interior starburst pattern. Sometimes the flesh may be yellow, brownish or off-white. Sweet tart with a slightly acidic edge, this decorative fruit’s succulent flavor is mainly sweet.

After intersecting the Fruits 360 data set with the descriptions available on specialtyproduce.com, I narrowed down 90 distinct fruit classes

I randomly chose 5 fruits to hold out as test fruits whose definitions/images would never be seen by the system: banana, clementine, eggplant, red onion, and red pepper.

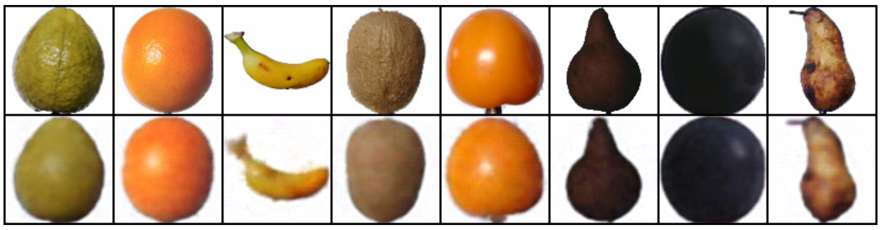
Figure 1: Reconstructions from the Image2Image ConvNet

3 Architecture and training procedure

3.1 Image2Image ConvNET VAE

I began by setting up the ConvNET VAE from the aforementioned introduction to VAEs. The picture below shows the quality of its reconstructions after 11 epochs of training. The only modification was I reduced the hidden dimension size to 512 in order to prevent overfitting over the text definitions.

3.2 Image classifier

I trained a ConvNet image classifier over my 90 fruits based on the code from the pytorch classifier tutorial.. The classifier was used to assess the quality of output images and factored into the loss function for the final Text2Image VAE.

3.3 Text encoding

I needed to design a method to represent the descriptions in a way that is suitable for input into a VAE. Initial embeddings I tried included custom RNNs and infersent embeddings but I found these to result in tremendous overfitting and a poor ability to generalize to unseen definitions. Eventually, I found that the best approach to avoid overfitting on the descriptions was a BoW representation ignoring all unigrams and bigrams save for a small hand-chosen set of 145 visually descriptive unigrams and bigrams. The BoW representation utilized td-idf weighting for each term in the vocab.

In addition to a 145-length vector describing the textual content, I also chose a set of 400 random 5-length vectors and assigned one to each definition/image pair. The motivation was that there is only one definition per class but potentially 100s of photos, all in different poses. That said, given that the noise vectors didn't parametrize pose space in any meaningful way, I decided to draw from only a small set of 400 vectors across all training and testing instances. This was to ensure that any random noise vector encountered in a testing instance would be guaranteed to have been seen in the test set - allowing us to exploit the overfitting on the training to associate each vector with some fixed set of poses. I chose 400 because each class had approximately 200 images, and I wanted vectors to be relatively unlikely to be chosen twice for two instances of the same fruit, but still likely to appear many times in the data set.

The text encoding portion consisted of 4 fully connected layers followed by two parallel fully connected layers one of which maps to a 256-length mean vector and another to a 256-length log variance vector. These are used for the parametrization of the VAE's latent space. Its input size is 150.

3.3.1 Pretraining

Rather than connecting the text encoding module directly a to a ConvNet decoder, I trained the text encoder to predict the mu and logvar vectors generated by the Image2Image encoder discussed in section 3.1. I trained it for 500 epochs. Test loss was evaluated over definition/image pairs for the 5 fruits whose definitions were excluded from the outset.

This pretraining helped to initialize the text encoder much closer to the ideal encoder than it would've been otherwise. Below you can see the results of plugging the pretrained text encoder into the pretrained decoder from 3.1.

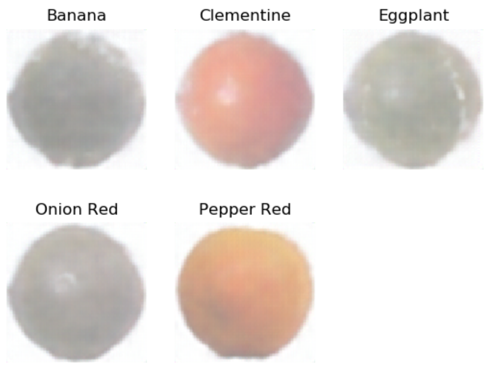

Figure 2: Decodings of text encodings following pretraining step

3.4 Text2Image VAE

Finally, I connected the pretrained text encoder from 3.3 into the decoder from 3.1 and trained it. I used a custom loss function $MSE + DISCRIM/200$, where MSE is the mean-squared error loss between the output image and the input image associated with this particular definition vector/noise vector instance. DISCRIM refers to the cross entropy between the class predictions from the classifier from section 3.2 and the true class identity of the definition/image.

Note that I omitted the KL-divergence term typically associated with VAE losses. This is because, due to the pretraining from sections 3.1 and 3.3 the combined encoder/decoder set up already had a contiguous latent space.

4 Results

I trained the resulting Text2Image VAE for 350 epochs. Below you can see the loss graph

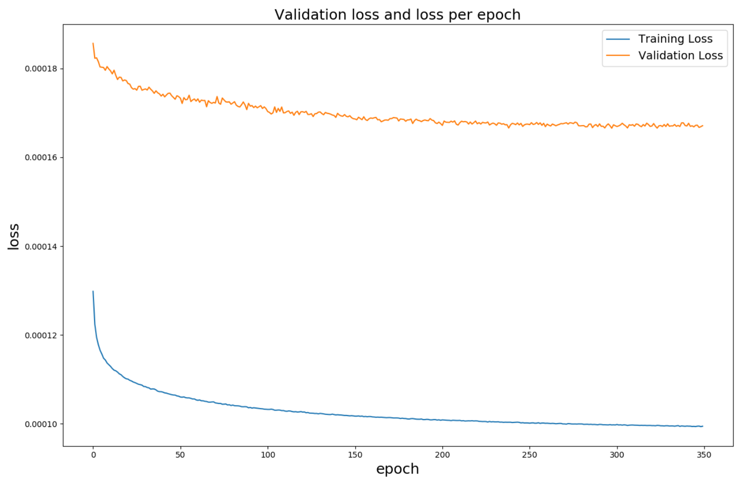

Figure 3: Training loss for Text2Image VAE

Below are the reconstructions from fruits in the training set

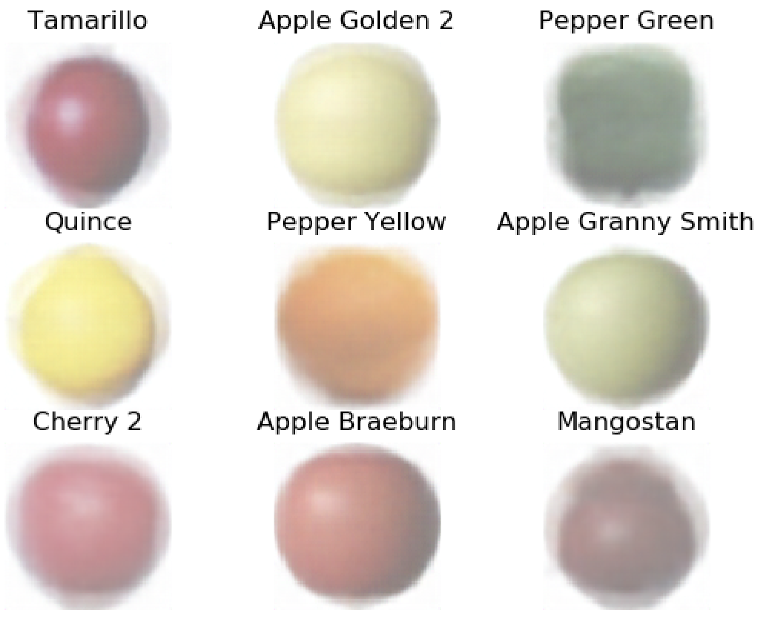

Figure 4: Test fruits generated by best Text2Image VAE

As you can see, the training fruits result in fairly faithful reconstructions with some blurriness resulting from the the variance in pose possible. Though the the pose and shape are better handled by the random vectors than by no vector at all, one might be able to see substantial improvement from specifically parametrizing the pose space in a meaningful way.

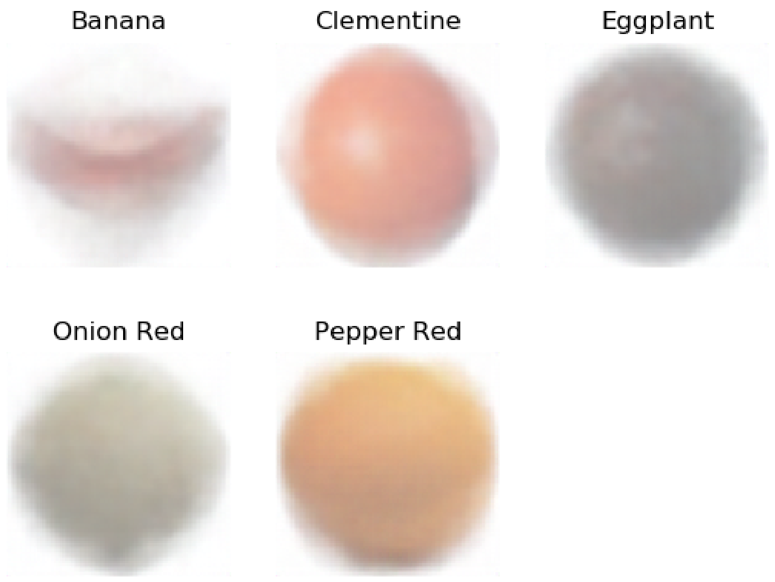

Figure 5: Test fruits generated by best Text2Image VAE for noise-free embeddings

The results on the test fruits demonstrate that the system has some ability to generalize to unseen text descriptions.

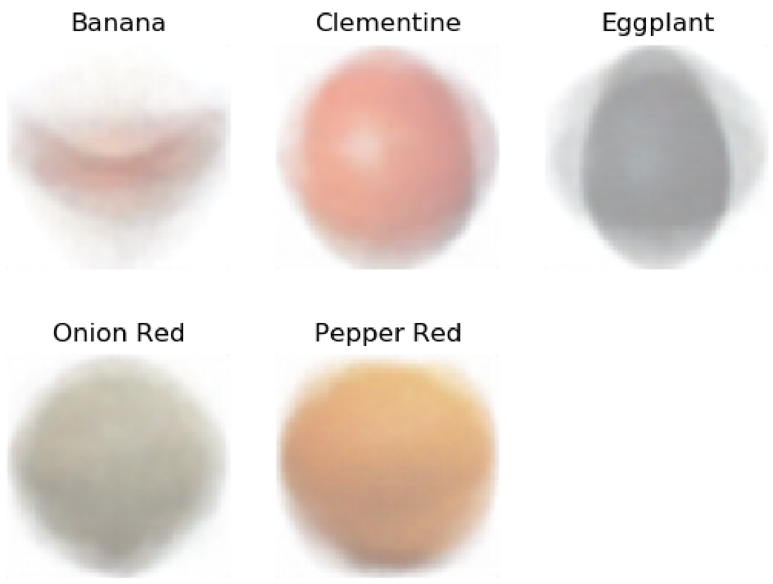

Figure 6: Test fruits generated by best Text2Image VAE

The banana comes out appearing red because the most similar fruit in the training set was the red banana. Additionally, the red pepper comes out looking yellow because the most similar fruit in the training was the yellow pepper. The eggplant also comes out looking like an avocado due to the peculiarity of eggplants and their similarity to avocados.

The fact that the unseen definitions tend to get projected onto the relatively proximate definitions from the training set shows that this system is promising and could benefit substantially from increasing the number of classes available.

Below you can see the results of generating images from novel text descriptions.

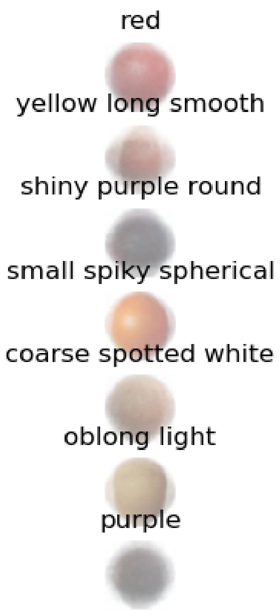

Figure 7: Fruits generated from unseen text descriptions

The system shows at least a basic awareness of the colors red and purple, as well as some understanding of lightness and oblongness. Otherwise it seems like the other features probed aren't learned at any level of granularity.

Below you can see interpolation between the descriptions 'red' and 'purple' and between 'flat' and 'long'. Rather encouragingly, the system is able to learn meaningful representations for these one word concepts.

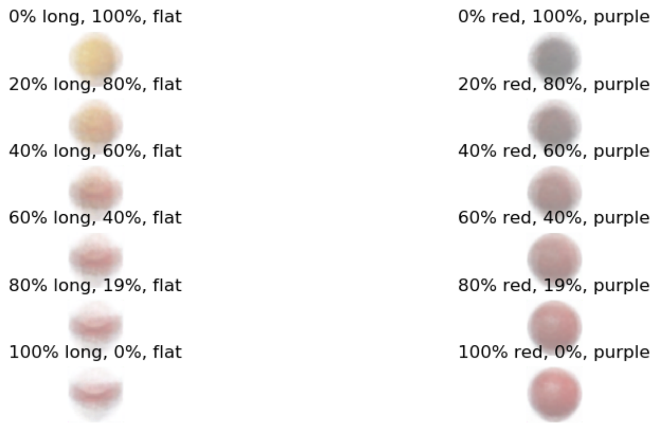

Figure 8: Images generated by producing the latent vectors for the two descriptions and interpolating between them

In an attempt to boost generalization and compensate for the dearth of textual data, I attempted to split the definitions into 3 sentence chunks and treat each as its own definition. Unfortunately it resulted in substantial underperformance, as demonstrated by the test fruits below:

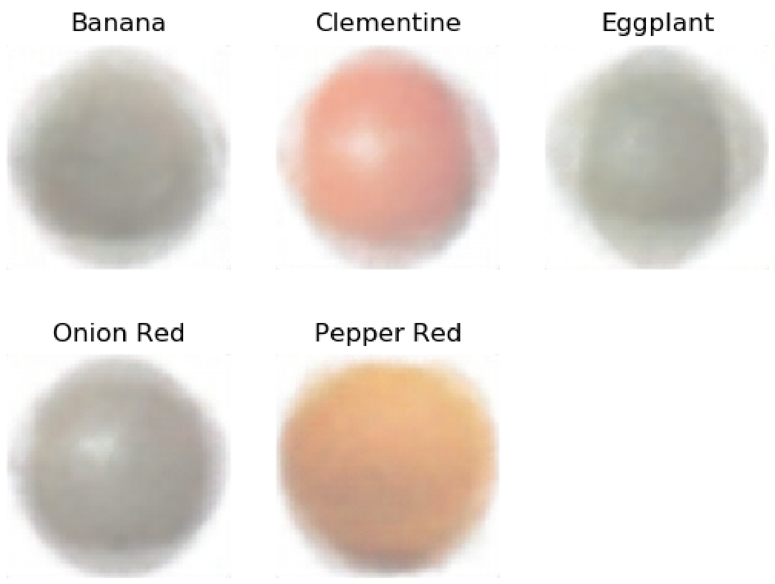

Figure 9: Fruits generated from unseen text descriptions

5 Future directions

Some very obvious shortcomings of the study are the number of object classes available was very small (only 85 text descriptions were seen during training), the content of the text descriptions (which were sourced from the open web and not tailored specifically for visual description tasks), and the paremetrization of poses.

An additional interesting route may be to integrate this work with conditional VAEs which condition the output based on additional information (in conjunction with the latent vector). For example, the link here uses the Celeb A data set to build a conditional VAE conditioned on 40 binary attributes (has_glasses, has_bangs, etc.).

Grounding text definitions in terms of attribute vectors strikes at the heart of intensional semantics and may be a very fruitful avenue for generating visual outputs from definitions alone.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[2] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

[3] Andrei Barbu, Alexander Bridge, Zachary Burchill, Dan Coroian, Sven J. Dickinson, Sanja Fidler, Aaron Michaux, Sam Mussman, Siddharth Narayanaswamy, Dhaval Salvi, Lara Schmidt, Jiangnan Shangguan, Jeffrey Mark Siskind, Jarrell W. Waggoner, Song Wang, Jinlian Wei, Yifan Yin, and Zhiqi Zhang. Video in sentences out. *CoRR*, abs/1204.2742, 2012.

[4] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *CoRR*, abs/1605.05396, 2016.

[5] Horea Mureșan and Mihai Oltean. Fruit recognition from images using deep learning. *Acta Universitatis Sapientiae, Informatica*, 10:26–42, 06 2018.