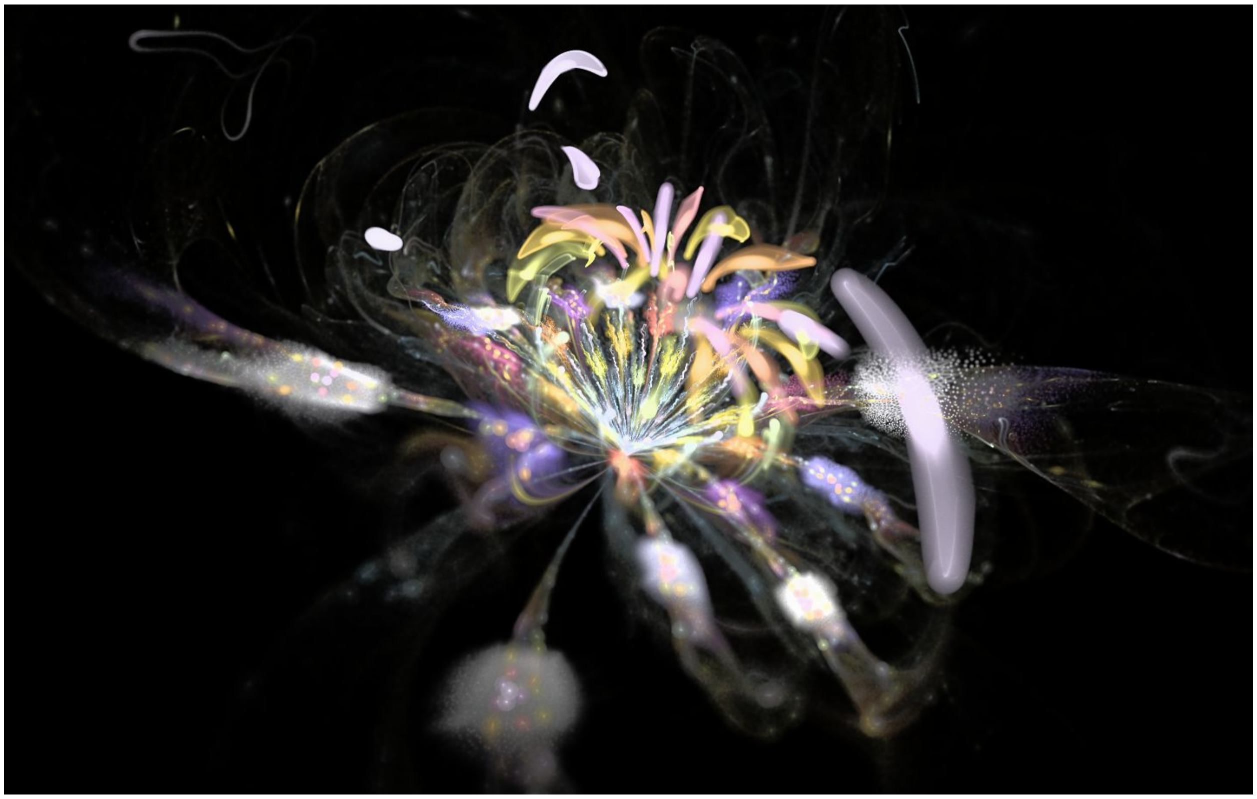
ART AND MACHINE LEARNING
CMU 2019 SPRING
FINAL PROJECT

Bloom: Visualizing Model Training through Fractals



Jean-Baptiste Lamare, Teven Le Scao, Craig Stewart, Liz Salesky, Nikolai Vogler

DESCRIPTION

Concept:

Neural networks have gained prominence for many tasks due to their often state-of-the-art performance, but understanding model behavior and the training process can be very difficult. For machine translation for example, interpretation and visualization of the model typically involve comparisons generated translations between small samples of sentences using either attention matrices or text comparisons or graphs of the model loss. Understanding trends across the dataset as a whole and the overall learning process, however, are very difficult to visualize. We here propose a visual, artistic way to better understand two aspects of our model training by conditioning fractal generation based on two aspects of our model training. We present a video of gradual fractal generation representing the model training process. First, we use the changing distribution of the embeddings for our target language is shown in the changing colors of our fractals, and second, the degree to which our fractal changes each frame is conditioned on the size of the optimizer step each batch in our translation model.

This project builds up on an ideas we have explored in the past (fractal generation in Project 2), but the techniques we use here are novel and serve a new objective.

The technique we propose is very flexible one, and could easily be adapted to visualize other machine learning models and other parts of the learning process (e.g. the outputs, the gradients, the model confidence…)

Visualizing Machine Translation (MT) Model:

We wanted to find a visual way to interpret the training process of a machine translation model. Neural MT models are typically encoder-decoder networks which use randomly initialized embeddings for source and target vocabulary words. Over the course of model training and optimization, we observe that generated outputs are typically at first the most common vocabulary items from training, and over time the model learns to produce more varied, conditioned translations of source language sentences. This process can be difficult to visualize only seeing the generated target text and e.g. changes between only parts of a sentence each epoch. We wanted to use visuals (generating conditioned colored fractals) to better see the change in and magnitude of updates to the network and embeddings.

We trained an encoder-decoder model to translate from Spanish to English using the Fisher Spanish-English dataset, containing translated conversational speech[1]. Our model has four LSTM layers, bidirectional for the encoder and unidirectional for the decoder, optimized with SGD and batches of 64 sentences. We constrained the vocabulary size for both source and target to be similar to the number of colors previously used to generate fractals, using 256 byte-pair encoding operations[2]. This dataset has a smaller vocabulary of only ~20k unique words, as it is originally speech, and therefore this smaller number of subword units actually corresponds with high model performance. We are also able to use a smaller embedding

[1] https://joshua.incubator.apache.org/data/fisher-callhome-corpus/
[2] https://arxiv.org/abs/1508.07909

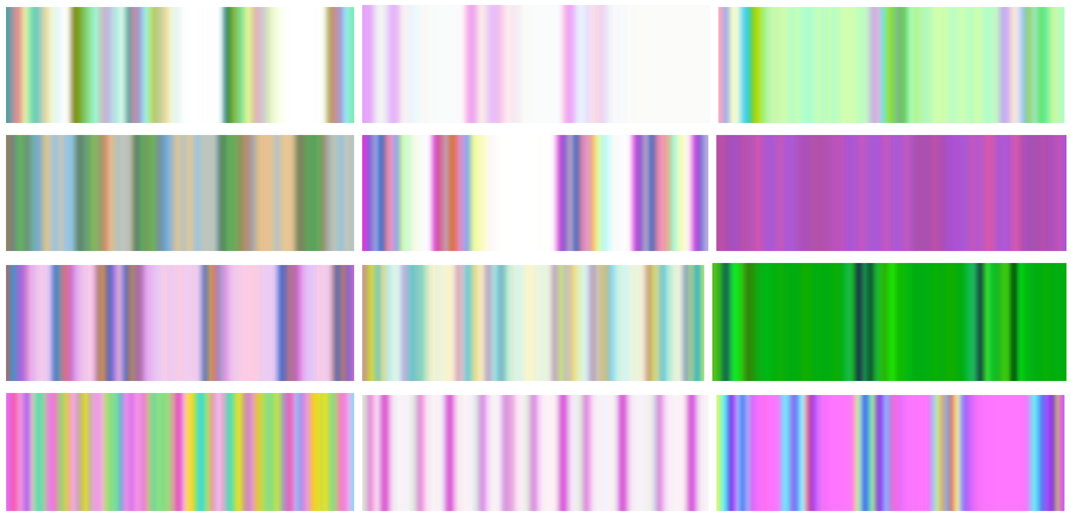
dimension because of this, with 64 dimensions, which further helps us in mapping the embeddings to a low-dimensional color space (discussed below). We chose this dataset in part due to the smaller vocabulary and consistent sentence structure, and therefore high-quality resulting translation performance, so that we would better be able to test our visualization method. We are able to produce translations with near-human quality (~60 BLEU)[3].

Mapping Embedding Changes to Colors

In order to inform the color of our fractal outputs, at each epoch of training of our MT system we extract the vocabulary of embeddings and perform dimensionality reduction thereon using Principal Component Analysis (PCA). This results in each embedding being represented in 3 dimensions.

We then sample random outputs and map the movement of the embedding vector during training to RGB color space. For example, a large shift in vector position in the first dimension would result in more red being applied to the corresponding color. We then stacked the color embeddings for each output across time.

For creative effect, we apply the following random interventions: seeding the color space, truncating the output lengths and flipping the color dimensions with a small probability.



[3] https://dl.acm.org/citation.cfm?id=1073135

Fractal Generation:

The final generated fractal is the fixed point of an iterated function system. We thus condition the fractal generation on the learning process by changing the dynamical system parameters based on the embeddings gradient: when updates are applied to the embedding weights, they are also applied to the system parameters, thus creating a continuously changing fractal.

Essentially, every function in the system is composed of an initial affine function that transforms linearly, then a a sum of non-linearities (the $k_i f_i$), then another affine mapping. A set of exactly one point is transformed into a set of n points by F ; if F satisfies certain properties in the set of the sets of points, then there exists a set X so that F(X) = X ; that set is the result fractal image. By varying the coefficients continuously, we can create a continuously moving fractal video. In order to do so, at every update step of the model, we add a dimensionally-reduced gradient update to the parameters of the fractal, thus making it evolve in line with the learning process of the model.

$$F(z) = \begin{cases} a_1(\sum_i k_{1i} f_{1i}(c_1 z + d_1)) + b_1 & \\ \vdots & \\ a_j(\sum_i k_{ji} f_{ji}(c_j z + d_j)) + b_j & \\ \vdots & z \in \mathbb{C} \\ a_n(\sum_i k_{ni} f_{ni}(c_n z + d_n)) + b_n & n \in [2, 10] \end{cases}$$

Process: Over the development of this project, we chose to limit the number of axes on which the fractal could change each iteration to more easily visualize changes. Indeed, as fractal objects tend to become degenerate and diverge numerically when the more sensitive parameters of the f_{ij} move, we've limited the updates to the affine transform parameters and to the colouring and opacity parameters.

Moreover, we use the color palettes described above to condition the color of the generated fractal, color that influences the way the shapes are drawn.

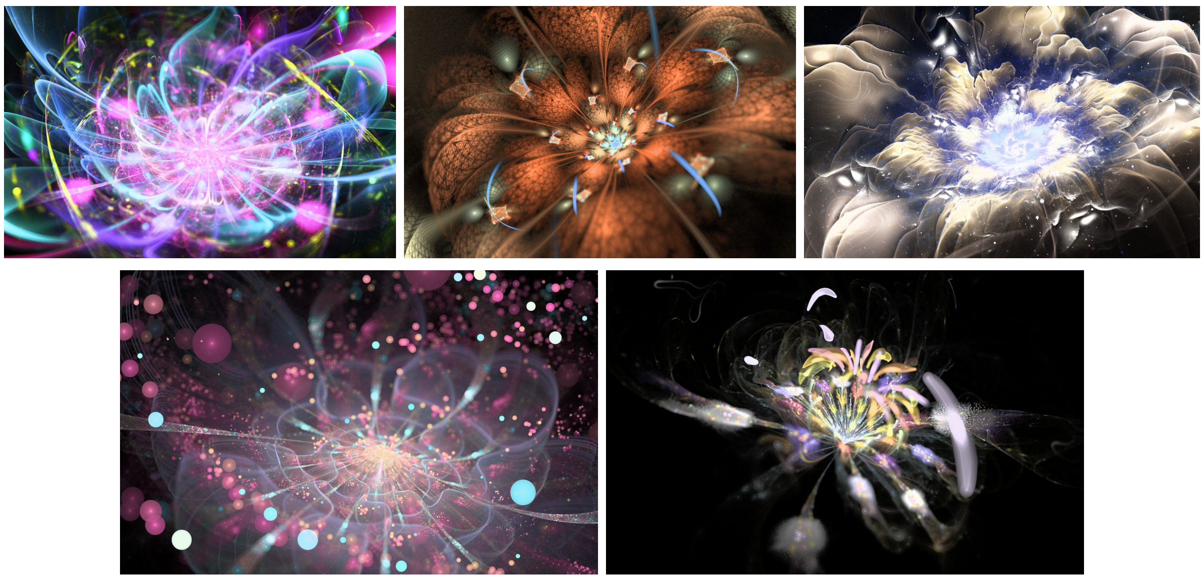
In experimenting with color palettes, we initially instigated a one-to-one mapping between the 3 dimensions of the reduced vector embedding and RGB values. Whilst this method was an interesting way of visualizing the color space occupied by embeddings we found that many of the vocabulary items were grouped in areas of shades of brown with minute variance between them. This was neither aesthetically pleasing, nor did it sufficiently capture our goal of showing the 'process' of training the embeddings.

As such we then opted to seed the color space at a randomized RGB point and represent the movement of embeddings by increasing or decreasing their corresponding RGB values. This created a much more satisfying result. Using temperature controls to exaggerate this movement we were able to travel much further across the color space for more varied effect. We then produced several color palettes from randomly sampled outputs and chose the most aesthetically pleasing of them for candidates for fractal generation.

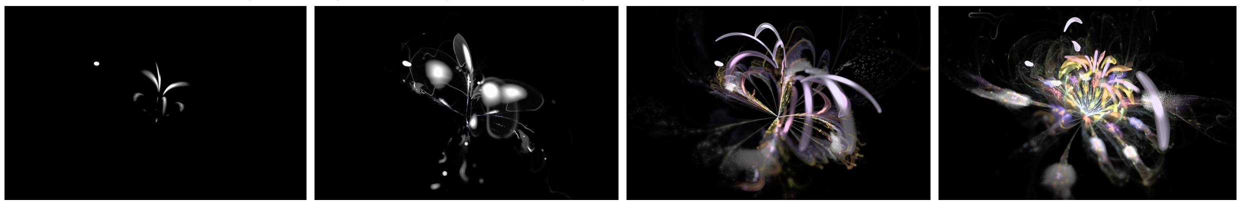
Given that some of the outputs appeared fairly uniform we also enforced a degree of creativity in palette generation in two ways. Firstly we randomly truncated the outputs along the time dimension such that the space between clusters of color was varied and interesting. Secondly, with a small probability, we randomly flipped the color dimensions from RGB to BGR. This had the effect of introducing a contrastive but complimentary color at random points in the palette.

Result: Our final result is a video with five different full model trainings, each conditioning generation of a different fractal. Stills from each fractal and showing the change in one over the course of model training are shown below.

Stills from individual fractals.



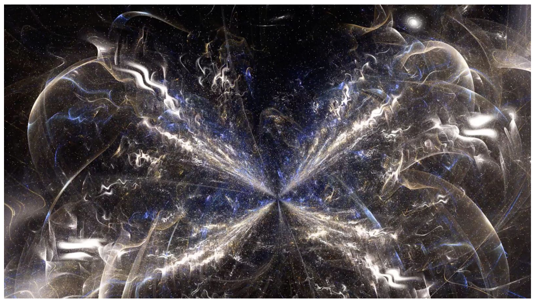
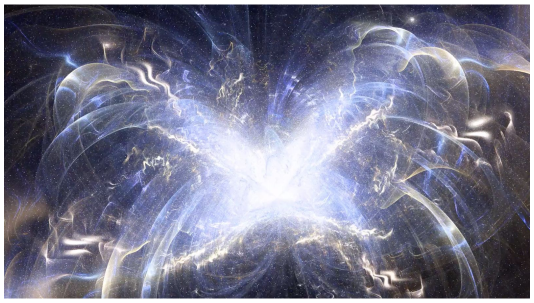
Stills showing change in single fractal generation over the course of model training.



We analyze two main aspects of our output, corresponding to the two aspects of our model we visualize: the change in fractal shape over time (corresponding to batch updates) and colors (corresponding to embedding changes).

We can see that at the beginning of training, the updates to the fractals small and consistent, because at this point in training, each batch has similar levels of error. The changes in the fractals frame-to-frame correspond to the scale of the update to the optimizer for each batch of sentences in training. As the translation outputs diversify and certain sentences are able to be better translated, we begin to see greater differences between generated frames, showing that the optimizer takes larger steps for some batches as compared to others. We note that the overall degree of change in the fractal generation is not linear over time: we see very small changes initially, and near epoch 5 as the model makes large changes and begins to generate more clearly conditioned translations we see more dramatic, rapid changes in the fractal. As the model converges, the changes in shape again become smaller. This information can be difficult to summarize across the full training process using plots, and we find our generated fractals to be a both visually pleasing and useful way to visualize convergence.

In each fractal generation, there are occasional examples where gradients explode, shown by a momentary large change in the generated fractal shape, as in this example:

We find this is much easier to visualize with our method than via text comparison or plots of model loss. There are also examples of jitter, where frame-to-frame the fractal seems to oscillate between similar states. This corresponds to those situations where different batches pull the optimizer in different directions, and again, we find this much easier to see visually with our fractals.

We use color to visualize the changes in the embedding space over the course of training. At the start of training, the embeddings for the target vocabulary are randomly initialized. When projected down to 3 dimensions, we see they are similar to each other in space, as shown by relatively monochromatic fractals. Over the course of training, the embeddings are updated through the translation loss, and while more similar words are pushed towards each other in vector space, the overall effect is that the embedding space diversifies. We can clearly see this through how more diverse color is introduced to our fractals over the course of training. The fractals are generated iteratively, with subsequent shapes building on previous. The colors are assigned to the iteratively introduced shapes from our generated color palette, with more diverse colors being used as subsequent shapes are produced.

Finally, we can see that the generated fractal stabilizes at the same time in training that the model converges to best performance.

Reflection: We are satisfied with our final result. The fractals are both visually pleasing, and are an effective way to view overall trends in the model training process and identify changes batch-to-batch. Additional suggestions which resulted from discussions at the exhibition include comparisons of generated fractals between multiple optimizers (e.g. SGD vs Adam), and creating a interface to aid in debugging where the fractal frames (each of which represent a batch) could be used to find those batches where e.g. the gradient explodes, which is easily visible here due to large changes between adjacent fractal frames.

Reference:

[1] Draves, Scott, and Erik Reckase. "The fractal flame algorithm." Forthcoming, available from http://IItlam3.com/flame.pdf (2003).

CODE: https://github.com/TevenLeScao/bloom

RESULT: The final video and selected high-resolution frames are attached in the submission email.