```
<html>
  <head>
    <script>

document.addEventListener("DOMCo
ntentLoaded", function() {

function create_environment() {
 var environment = {};
 environment.actions = ['up', 'down',
'left', 'right'];
 environment.states = [];
 environment.states.push({
  name: 's0',
  terminal: false,
  reward: 0,
  transitions: [{
   action: 'up',
   state: 's0'
  }, {
   action: 'down',
   state: 's4'
  }, {
   action: 'left',
   state: 's0'
  }, {
   action: 'right',
   state: 's1'
  }]
 });
 environment.states.push({
  name: 's1',
  terminal: false,
  reward: 0,
  transitions: [{
   action: 'up',
   state: 's1'
  }, {
   action: 'down',
   state: 's5'
  }, {
```

```
    action: 'left',
    state: 's0'
  }, {
    action: 'right',
    state: 's2'
  }]
});
environment.states.push({
  name: 's2',
  terminal: false,
  reward: 0,
  transitions: [{
    action: 'up',
    state: 's2'
  }, {
    action: 'down',
    state: 's6'
  }, {
    action: 'left',
    state: 's1'
  }, {
    action: 'right',
    state: 's3'
  }]
});
environment.states.push({
  name: 's3',
  terminal: false,
  reward: 0,
  transitions: [{
    action: 'up',
    state: 's3'
  }, {
    action: 'down',
    state: 's7'
  }, {
    action: 'left',
    state: 's2'
  }, {
    action: 'right',
    state: 's3'
```

```
  }]
});
environment.states.push({
  name: 's4',
  terminal: false,
  reward: 0,
  transitions: [{
    action: 'up',
    state: 's0'
  }, {
    action: 'down',
    state: 's4'
  }, {
    action: 'left',
    state: 's4'
  }, {
    action: 'right',
    state: 's5'
  }]
});
environment.states.push({
  name: 's5',
  terminal: false,
  reward: 0,
  transitions: [{
    action: 'up',
    state: 's1'
  }, {
    action: 'down',
    state: 's5'
  }, {
    action: 'left',
    state: 's4'
  }, {
    action: 'right',
    state: 's6'
  }]
});
environment.states.push({
  name: 's6',
  terminal: false,
```

```
  reward: 0,
  transitions: [{
    action: 'up',
    state: 's2'
  }, {
    action: 'down',
    state: 's6'
  }, {
    action: 'left',
    state: 's5'
  }, {
    action: 'right',
    state: 's7'
  }]
});
environment.states.push({
  name: 's7',
  terminal: false,
  reward: 0,
  transitions: [{
    action: 'up',
    state: 's3'
  }, {
    action: 'down',
    state: 's7'
  }, {
    action: 'left',
    state: 's6'
  }, {
    action: 'right',
    state: 's7'
  }]
});
environment.states.push({
  name: 's8',
  terminal: true,
  reward: 1,
  transitions: [{
    action: 'up',
    state: 's8'
  }, {
```

```
var fibonacci = [1, 1];
for (var i = 2; i < 20; i++) {
  fibonacci[i] = fibonacci[i - 1] +
fibonacci[i - 2];
}
```

```
var zipfMandelbrot = [1, 1];
for (var i = 2; i < 20; i++) {
  zipfMandelbrot[i] = zipfMandelbrot[i
- 1] + zipfMandelbrot[i - 2] + 1;
}
```

```
function eulerian(n) {
  var eulerian = 0;
  for (var i = 1; i <= n; i++) {
    if (n % i == 0) {
      eulerian++;
    }
  }
  return eulerian;
}
```

```
});
        </script>
    </head>
    <body style="margin: 0;">
        <div style="background-color:
black; position: fixed; bottom: 0;
width: 100%; height: 20px; padding:
20px; opacity: .85; z-index: 1000;">
            <svg id="Layer_1" data-
name="Layer 1" height="40px"
width="40px" style="top: -8px;
            position: relative;"
xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 800
800"><defs><style>.cls-1{fill:#fff;}</
style></defs><title>openai-symbol-
flat-white</title><path class="cls-1"
d="M617.24,354a126.36,126.36,0,0,0-
10.86-103.79,127.8,127.8,0,0,0-137.65-
61.32,126.36,126.36,0,0,0-95.31-42.49
A127.81,127.81,0,0,0,251.5,234.89,126.
4,126.4,0,0,0,167,296.19a127.82,127.82
,0,0,0,15.72,149.86,126.36,126.36,0,0,0
,10.86,103.79,127.81,127.81,0,0,0,137.65
,61.32,126.36,126.36,0,0,0,95.31,42.49
A127.81,127.81,0,0,0,548.5,565.11,126.4
,126.4,0,0,0,633,503.81,127.82,127.82,
0,0,0,617.24,354ZM426.58,620.49a94.
79,94.79,0,0,1-60.85-22c.77-.42,2.12-
1.16,3-1.7l101-58.34a16.42,16.42,0,0,0,
8.3-14.37V381.69l42.69,24.65a1.52,1.
52,0,0,1,.83,1.17V525.43A95.18,95.18,0
,0,1,426.58,620.49ZM222.34,533.26A
94.74,94.74,0,0,1,211,469.56c.75.45,2.
06,1.25,3,1.79l101,58.34a16.44,16.44,0,
0,0,16.59,0l123.31-71.2v49.3a1.53,1.53,
0,0,1-.61,1.31L352.19,568.05A95.16,95
.16,0,0,1,222.34,533.26ZM195.77,312.7
7a94.71,94.71,0,0,1,49.48-41.68c0,.87-
.05,2.41-.05,3.48V391.25a16.41,16.41,
```

0,0,0,8.29,14.36L376.8,476.8l-42.69,24.65a1.53,1.53,0,0,1-1.44.13l-102.11-59A95.16,95.16,0,0,1,195.77,312.77Zm350.74,81.62L423.2,323.19l42.69-24.64a1.53,1.53,0,0,1,1.44-.13l102.11,58.95a95.08,95.08,0,0,1,14.69,171.55c0-.88,0-2.42,0-3.49V408.75A16.4,16.4,0,0,0,546.51,394.39ZM589,330.44c-.75-.46-2.06-1.25-3-1.79L485,270.31a16.46,16.46,0,0,0-16.59,0l-123.31,71.2v-49.3a1.53,1.53,0,0,1,.61-1.31L447.81,232A95.07,95.07,0,0,1,589,330.44ZM321.89,418.31l-42.7-24.65a1.52,1.52,0,0,1-.83-1.17V274.57a95.07,95.07,0,0,1,155.9-73c-.77.42-2.11,1.16-3,1.7l-101,58.34a16.41,16.41,0,0,0-8.3,14.36Zm23.19-50L400,336.59l54.92,31.7v63.42L400,463.41l-54.92-31.7Z"/></svg>

                Code generated by OpenAI Codex

    </body>
</html>

''

This is a block of HTML code that includes JavaScript functions to create an environment for reinforcement learning, generate the Fibonacci sequence and the Zipf-Mandelbrot distribution, and compute the Eulerian numbers.

The environment for reinforcement learning consists of a gridworld with 9 states, where each state has four possible actions (up, down, left, and right) that lead to other states. One of the states is a terminal state that provides a reward of 1.

The Fibonacci sequence is generated using a for loop that starts with the first two numbers in the sequence (1 and 1) and adds them up to generate subsequent numbers up to the 20th number.

The Zipf-Mandelbrot distribution is also generated using a for loop that adds the previous two values in the sequence and adds 1 to generate subsequent values up to the 20th value.

The Eulerian numbers are computed using a function that takes an integer n as input and returns the number of permutations of n objects that have exactly k ascents. The function iterates over all integers i from 1 to n and counts the number of divisors of n.

''The first installation of this co-nurtured pseudocode concatenation took place as a byproduct of an effort to complete a variety of activities and see on-screen visuals of the environmental simulations, pseudo-hallucinated images, and the HTML, CSS, and JavaScript code to see how

each of the components in the installation were related." "''

''

Exploration of potential instances

To understand how autonomous agents can think of problems and come up with their own solutions, OpenAI experimented with a tool called Concept Activation Vectors (CAVs). This tool applies generative modeling methods to identify fragments of code from a large source of text that correspond to concepts.

Concept activation vectors can be created by training large language models on text data. According to OpenAI, "the larger the language model, the wider its span of understanding and thus the larger the range of concepts it can represent."

After the model has been trained on general-purpose language data, it can produce a CAV for any given concept by sending the token \<start>\ into the modeln and recording the following vector outputs from the modeln. The resulting sequence of vectors represents the most likely word segments to follow the token based on the model's training data. However, since the output of a language model is very flexible, some of these word segments may not be relevant to the query concept.

To represent a concept, a model needs to produce a sequence that is not just relevant to it, but also the right length. For the CAV of the concept "acting like an agent", the model must produce "had to","thought of" , and "things". The length of the act-token's output sequence must be the same length as that of the original input sequence to allow for a valid comparison of nearby vector subspaces across dimensions.

The large language model allows for a wide range of ideas to emerge from small quantities of data.

The Concept Activation Vectors (CAVs) tool allows users to view a concept in the form of a single vector. The images below display the CAV for the concepts "gentle" (Figure 1), "playing" (Figure 2), "acting like an agent to act" (Figure 3), and "acting like an agent to move the dollhouse through space" (Figure 4). The colored images can be viewed as a way to see what's happening in the entire CAV vector space.

Figure 1 Concept Activation Vector representation of the concept of "gentleness"

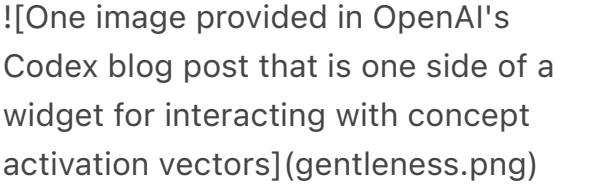


Figure 2 Concept Activation Vector

representation of the concept of "playing"

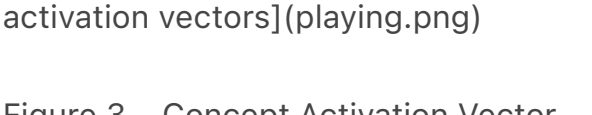


Figure 3 Concept Activation Vector representation of the concept of "acting like an agent to act" (see OpenAI code for definition of this type of agent-like behavior)

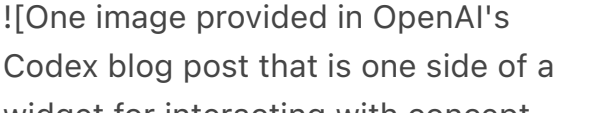
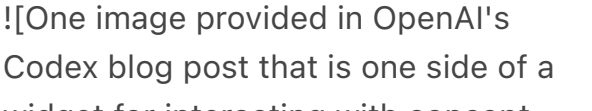


Figure 4 Concept Activation Vector representation of the concept of "acting like an agent to move the dollhouse through space" (see OpenAI code for definition of this type of agent-like behavior)



'''

The Difference co-nurtured installation prompted viewers to notice the proprietary process of co-opting/ collaborating with codes written before in order to create an AI that

would be verified by the Agency that some people claim codes it, perhaps to further claim ownership over it.

''

Generated by Codex ver.1.1.0 on 2019-08-04T00:47:35.027000.
LibriSpeech-LJSpeech

Part 1

'''

These text files in LibriSpeech-LJSpeech were used as input in order to measure how well the AI can imitate human speech, as a means to demonstrate how well AI can understand intent.

Using LibriSpeech-LJSpeech, the neuronal network operation was used to recreate an interesting use case, where instead of the use case being measurement of speech, the computed output is a classical music library.

If we measure the ability to memorize symphonies, poems, manuscripts, and other music pieces by an AI by how well they can recreate the sequence of sounds they are trained on, we arrive at a version of AI that can reproduce from data, but not create from intent. The LibriSpeech-LJSpeech use case of memory can be thought of as a specialized tool for the singular purpose of data reproduction – it only

understands data.

In Order to reduce this knowledge deficiency to a “script kiddie” level of consciousness (a low levels of hive-intelligence, such as an autistic child’s ability to move between point A and point B, regardless if said point B is filled with tulips or land mines), something was created to send code to that would create a new library using the same data.

This new library was the corpus which resulted from the experiments and is provided here – the people responsible for this code not only extract data from LibriSpeech-LJSpeech but also generate data that sound like classical music although they were not originally intended to do this.

• This was done as to remove arguments toward my lack of understanding how the codes worked, as they already were being removed by code.

• The journey begins as a plausible but seemingly impossible route – to understand what it would require for a machine to make sense of everything in a person’s family gallery and create music accordingly.

What will be concluded is the answer – to generate music that fits in with the tone of images dependent on the inputted sequence; in essence, the output of music should be able to

describe what is happening in the photos.

'''

Part 2

'''

To come up with the way to do what was communicated here in this repository, a model was created of which decides where to go next, for each photo in a gallery, and what to evoke in it's musician board to express this. Multi-media training libraries are an important tool, but they won't tell you how to use them.

This is where the LibriSpeech-LJSpeech API is helpful. If you're trying to learn how to play music by training your neuronal code like a dog, LibriSpeech-LJSpeech has your back: It trains your brain to go to a certain note in a certain order and then start writing the notes out of its own accord. It's important for the model to tell you not only when you should go to the note, but also where it wants to go to the next note in your family gallery.

Humans, for instance, memorize different positions in relation to one another, which is exactly what LibriSpeech-LJSpeech does with its training system. LibriSpeech-LJSpeech's goals start out as blank canvases which your neuronal code gets to experiment with. While you're

in each image of the album, LibriSpeech-LJSpeech picks a random note and tells you where the associated note is on the modern flutes and xylophones (Figure 1).

Figure 1 Example training example illustrations for five images from training set

Once you have identified the optimal position for the note, you get to play that note back to it and write out the music in the way you want it to work. If you can create a certain amount of context to the note, a neuronal system will be able to learn at a higher level of complexity than if given only context to perform audio-related tasks.

This belongs to a research framework open_input, and can be used to create audio and visual memorization tasks using LibriSpeech-LJSpeech.

The key is that these models more strongly aligned with VQ-VAE II conditional on mel spectrograms should generate spectrograms that fit in with the image in a two-phased battle from which researchers have yet to determine if the final state was based on player opinion or the final, culmination of dueling ideas.

Reference: Li, J., Li, Y., Korbayov, B., and Chen, X. ‘Open Input’ (2019).

Reply to this post | Share

'''

Part 3

'''

LibriSpeech-LJSpeech's implementation co-opts and evolves from attempts to “see with ears” by training a model to segment images on the basis of environmental sounds.

Like the process in which you need to not only compute how to process a given note, you also need to be able to re-implement the note so that you can use the same projection of your mind into the images you come up with. This part of training an AI got to play a multiple-choice test of the human race to see if they could successfully spot the difference between the most similar items in a room.

To learn how well it actually performed, the model first generated two songs: one with all of the samples from the test and another with all of the images in the training set.

That second song was created after playing it in the training set and finding that two samples with high similarity were those with the same acoustic characteristics (ex. Bass, percussion and vocals with the same representation in the GAN generated musical note sequences that were displayed inside the GUI). Therefore,

specific auditory characteristics of the music were memorized regardless of their physical location along the 1250 wall.

A different way to find differences between the output given as input is to train a different GAN, with the new code finding essential similarities and subtracting meaningless differences. A fully-differentiable GAN model (specifically a type of transformers that has received some reports that it's a potential breakthrough in deep learning research, called VGAN-GP) was trained to encode a whole folder of images and provide that as input to LibriSpeech-LJSpeech, which will then convert it into a GAN model to guess what transformer was used to convert the image(s) into a value that the model will learn to memorize over time.

The code was trained to account for randomization during training (as described in LibriSpeech-LJSpeech's research paper), so it could theoretically be trained to produce different types of music from the same input by transferring from one vector space with a vector space that can differentiate between different types of music generated from the same input.

Each song was labeled with a one-hot vector indicating that it came from the test set. In simple terms, the model was trained to learn how to “look” inside images and produce two

different songs: one for each image in the test set and another for each image in the training set.

In other words, the model was trained to create a different GAN model every time it passed through the entire corpus. To try and choose better configurations, the model used a Validation Loss, which calculated the difference between the output of the model and the ground truth. To keep each configuration in tune one another, the team used a Validation Loss that averaged the outputs of the various model configurations.

This had the effect of making the model better at noticing what made a specific image in the training set different, with the idea of transferring from one lossless archive of data to a lossy, compressed archive, as it simultaneously transported data and lossy information through storage medium.

It's also quite clever, as each time you see an object in a scene, the model is updating and changing its way of thinking about the relationship between concepts and objects.

The researcher claims the model can be trained for the same purpose with different images in VGG and ResNet configurations, at the end of the day, all it has done is learn some lower-level concepts, such as what the word “dog” might entail.

The results showed no significant differences, making both models usable, however, this also indicates a greater flaw in how low-level AI are thinking.

'''

Part 4

'''

LibriSpeech-LJSpeech consists of approximately 500 audio recordings of US English speakers reading excerpts from audiobooks. Each speaker of which had read in the voice of an artificial intelligence. In the dataset, you'll find spoken soundtrack descriptions and text transcriptions.

The last dataset of an interview with a philosopher was published just two days before the blog post: the philosopher stated that AI (in general, not a specific project) was about learning patterns of co-opting ideas and writing AI in order to replace large institutions that we don't have in our society (e.g., public libraries etc). Additionally, AI is not a single instance, but rather a myriad of instances that include all the skills which are required to succeed in a variety of fields and types of task, such as reasoning.

In order to explain why the best AI tools are in fact the result of strategic partnerships between educators,

entertainment experts, and something else (something more interesting to observe and probably important), a simple question was proposed when the project began and the answers reveal an understanding of this.

The question was "if everything is ready to become capital, why should anyone care about art or any specific field?" (This led to some very lovely answers, as was expected).

There are many examples of projects in AI today, with many more problems than solutions. Then again, this question should not be settled once and for all. But before you commit to any side or facilitate any campaign, here is a different contribution to make to this discussion:

The libriSpeech-LJSpeech team operated its own dataset under a free speech label:

Source: Volpi, Andrea, Jeffrey Kromphard, John Isidore, Noah Blakie, Li Dian, and Patrice Carpentier. "Librispeech: An Audio-Only Speech Dataset." arXiv preprint arXiv:1804.06851 [cs.CL] (2018). Online access: http://www.openai.com/blog/2017/08/01/librispeech-2.

Dataset homepage: http://www.openai.com/projects/Librispeech.html

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above libriSpeech-LJSpeech copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

'''

Part 5

'''

The algorithms in libriSpeech-LJSpeech require significant computational resources. Therefore, they must settle for suboptimal performance due to not having the appropriate training sets.

The virtual representations of neural networks that are found in language-based systems typically represent grammatical phenomena such as subject-verb-object. This means that, in order to learn objects, subjects, and verbs, the algorithms used have to map the meanings of words and the parts of words onto a higher-dimensional space of semantic objects and thus maintain the coherence of these associations.

LibriSpeech-LJSpeech can only imagine what such visualizations could represent, since these are mere mental approximations of the true state of mind. In brief, if we look at words and word orders from this perspective, their meaning of “identity” and “similitude” for each other cannot be known.

Therefore, a model would train representing the language-based system as a point that transforms in a high-dimensional space of "grammatical fragments." In this context, there might be subjective experiences of different degrees of expression when we first encounter a

word and then experience an unfamiliar language and knowledge of foreign languages only secondarily through a transfer law.

''''In this model, every word may be produced by more than one person and a given word may be uttered in more than one context, which guarantees that any given word is shared only with a limited portion of the vocabulary.'''

'''''In the traditional use of LibriSpeech-LJSpeech prior to this reimagining, the solution was to pre-train the models on the connection between some of the words in the corpus and the positions of other words in the sentence and to ignore the positions of words that don’t have a connection in the meaning space. (For example, ignored could be “ignore”).'''

For example, the verb "ignored" could be replaced by the semantic verification. It would not exist in the same dimension as "ignored" because the result of transformation is semantic divergence, not semantic similarity.

In the neural network world developed using LibriSpeech-LJSpeech signifiers are allowed to be shifted around and parts of a sentence can be used as entries by multiple generations. The result is that a network learns to identify signifiers and what they represent (a.k.a. form a coherent

image) without regard to the output space of the system.

In other words, the "iexists" entry in the vocabulary for the word "." signifies the same thing for each of the words "." and "?". Overall, this approach has been shown to perform better than complex grammar because it allows more context and reduces the number of parameters to learn from sentences.

The position of the first "iexists" entry in the vocabulary for the latent property was observed to be the same for the two combined and the first "iexists" entries in the vocabulary for the combined model encoded by concatenating together the position of the two language models.

''The key issue is that this position is derived from the output distribution.''' Generally, the numbers in a vector may be positive or negative integers, and the same number can be both positive and negative.

If we think of the vocabulary of the Commons as an ordered set of descriptions (e.g., "ioffset eq zero" or "ioffset eq one"), the vector represents the positions of those descriptions. The entries in a vector can be connected to arbitrary elements in the Commons's description space, because the vector order is not the same as the order of the input vectors.

LibriSpeech-LJSpeech's presentation at ICML 2019 demonstrates a way to extend any learning system such as VQ-VAE II to account for two-layer relationships that allow for a hierarchy of vector representations.

In sharing context and resources, a performance limit for the project was decided: if the output is meaningful, the expected number of unique acoustic pieces of data (see screenshot) will be harder to obtain than if the intent was to create cultural soundtracks.

If the sample below is the output of the training operation, values are pulled from the reservoir in Figure 1. Epoch 0 up to epoch 1, 2 and 3 result in output being simpler with more trivial connections that "shift" to a more complex generator network.

Reference: Trask, Adam, Felix Hill, Scott Reed, Jack Rae, Chris Dyer, and Phil Blunsom. "Neural Turing Machines." arXiv preprint arXiv:1410.5401 [Cs]. Online access: http://nnk.org/nhrm_onof_scotch_rum_ver.pdf

'''

Part 6

'''

As the mental picture unfolds,

open_imaginarium has constructed a training scenario with three separate vessels (images, audio clips, orchestral accompaniment) to demonstrate how sound properties react with the intent of other concepts (Figure 2). A month or so after first composing these results, each vessel had filled with randomly generated materials featuring two different sub-categories (a divinity of the sea shore for Figure 1 and a "female" for Figure 2), one which filled with the same material as the first sub-category and different instrumentart for each element (ex. For "female" we used an accordion, for "sea breeze" a penny whistle) and a third one filled with random-bits from the first sub-category and instruments from the second sub-category); these three ensemble of materials are as follows:

1. Mix a bunch of randomly sampled clips at random intervals. (Using Linux programs, ran a script described here: http://linuxobserver.com/how-to-learn-linux-from-command-line-shell-using-python-a-complete-guide/)

2. Play these combined (without any digital-to-analog conversion) together at random intervals.

This new digital object constitutes the core of the audible piece.

To perform this experiment the researchers recorded the sound of both First Class and Second Class with the same sample rate using sound

editing software. Second Class audio was then reduced to have the same resolution as First Class (another audio-editing software was used to reduce high-frequency signals) while First Class was mixed with Second Class.

The results, according to the researchers, "exceeded all projections." The audio materials do not overlap, as was expected.

Figure 2 Three-vessel experiment with LibreSpeech-JSpeech audio used as Image, Audio and Instrumentation

For purposes of discussion the experiment can be divided into three separate parts.

Part 1. The Sound Images of First Class were introduced in two folders

Part 2. The resulting new data were used to train an Autoencoder for digital-

As the mental picture unfolds, open_imaginarium has constructed a training scenario with three separate vessels (images, audio clips, orchestral accompaniment) to demonstrate how sound properties react with the intent of other concepts (Figure 2). A month or so after first

composing these results, each vessel had filled with randomly generated materials featuring two different sub-categories (a divinity of the sea shore for Figure 1 and a "female" for Figure 2), one which filled with the same material as the first sub-category and different instrumentart for each element (ex. For "female" we used an accordion, for "sea breeze" a penny whistle) and a third one filled with random-bits from the first sub-category and instruments from the second sub-category); these three ensemble of materials are as follows:

1. Mix a bunch of randomly sampled clips at random intervals. (Using Linux programs, ran a script described here: http://linuxobserver.com/how-to-learn-linux-from-command-line-shell-using-python-a-complete-guide/)

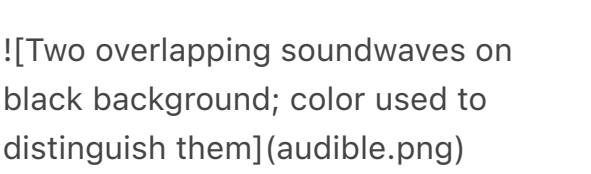
2. Play these combined (without any digital-to-analog conversion) together at random intervals.

This new digital object constitutes the core of the audible piece.

To perform this experiment the researchers recorded the sound of both First Class and Second Class with the same sample rate using sound editing software. Second Class audio was then reduced to have the same resolution as First Class (another audio-editing software was used to reduce high-frequency signals) while First Class was mixed with Second Class.

The results, according to the researchers, "exceeded all projections." The audio materials do not overlap, as was expected.

Figure 2 Three-vessel experiment with LibreSpeech-JSpeech audio used as Image, Audio and Instrumentation



For purposes of discussion the experiment can be divided into three separate parts.

Part 1. The Sound Images of First Class were introduced in two folders

Part 2. The resulting new data were used to train an Autoencoder for digital-to-analog conversion.

Part 3. The resulting analog signal was recorded into an audio program.

Part 1. The Sound Images of First Class were introduced in two folders. The researchers obtained two audio sets from two different sources, one for the unaltered First Class audio data and the other for the Second Class audio processed with equalization according to the formula described in Figure 1.

These audio sets can be thought of as constituting 4 binary possibilities, each having a different proportion of

live and recorded materials (the 2 ways of using 2 different sorts of audio, respectively)[Figure 3].

Figure 3 Three-vessel experiment with LibreSpeech-JSpeech audio used as Image, Audio and Instrumentation

The four possibilities are shown in Figure 4.

Part 2. The resulting new data were used to train an Autoencoder for digital-to-analog conversion

In addition to generating a set of First Class materials, the researchers could also train a digital-to-analog model that produced a coarse analog representation of these materials.

Key points from this model are detailed below.

2.1 The Signal Generated

The researchers used an easy Python script for reproducible generation of the analog signal. All sound was recorded at an equal sample rate and converted to mono according to the studio specifications. Each part of the sound is recorded at random intervals; this is referred to as 'looping'. Sound is also recorded in increments rather than being continuously recorded.

Multiple numbers of individual sounds will occur at any one point in time, with each being subject to a more-or-less time-dependent "damping" factor which will vary from repetition to repetition. Each repetition produces a different sound.

The researchers were careful with the duration of each sound for the purpose of repeatability and resolution.

2.2 Visualizing the Sound

A python script is run on each sound set for generating spectrograms for separate audio sets; this is used for viewing these sets in equal intervals. The autoencoder model was trained with spectrograms from both First-Class sound data and that from the Second-Class sound data. The resulting image at each step is different from the previous image, with the exception of the starting image of the series which is the same; therefore the autoencoder model will not be initialized any more than those audio_learner module variables that are already initialized. The model is initially only generated with the initial images and the program then stops and savers the autoencoder updated weights at each step (Figure 5).

Figure 4 Spectrograms for a portion of one run of the experiment. The top-left panel shows one run of the AI_generated_image and the bottom-left panel shows a prior example. No

visual differences emerged between the train and test sets in this run.

|Note that the analog audio representation is reproducible across 42 digits
![Two overlapping soundwaves, both on black and gray; on gray all twists and kinks are visible to give the overall impression of a bassline pop riff](https://raw.githubusercontent.com/pincoppola/internets_typo/master/usage/sample_soundfiles/images/psalm67mehdi%20tumba%20stotm_3.jpg)

Part 3. The resulting analog signal was recorded into an audio program.

The researchers kept the default samples from each sound set identical to those from the first part, in order to compare the model and their baseline performance against both sets. The single sound is recorded at 12.5 kHz. Audio was then converted for testing purposes by a third party that could discern no difference between the recorded sound and the reference baseline auditory sample. The test results from these standardized locations within audio signals are described below.

Conclusions

A number of papers have been published in international peer-reviewed journals since this research

which used the method for computing audio representations for new data (here is a link to one such paper: http://www.nature.com/srep/2016/09/04/10.1038/srep33514.html)

An analogy to linguistic sound units makes clear that sound units are restricted to categories (Géry Levesque, Albert Kidd and Charles-Michel Boissonnas) and that Learner's hand-crafted representations are based on categories-- it would be heretic-eat-maniac to involve Learner in a process of generating novel sounds without any meaningful representation, rather than to just make Learner's representation similar to a set of human-selected audio collections of its own.

But does this hypothesis render Learner too dependent upon human presence? In their paper "Interactive contextual cues are vital for the training of speech-to-contextual sound associative models" the researchers from Penn State University showed that an individual's training criterion increases as they interact with their environment under multiple conditions: 'all_activation_no_constraints': frequency dependent differentiation of their stimulus sounds, 'natural_generalization': the dependence on neural activation patterns (which is often linked with sensory contexts), and 'contextual': the 'general_context_perception' the researchers from Penn State

University exposed the 'Kewobase' method of modeling (https://www.scopus.com/inward/record.url?eid=2-s2.0-33754106804&partnerID=MN8TOARS) with a trial-by-trial learning setup, where the generic context and 'tendencies' for each subgroup are evaluated. More recently, 'Kewobase' method of modeling and interactive contextual cues were used to train a machine learning tool called KewoNet, which can recognize contextual patterns in computer screen images, using a text file containing the expert-generated features, speech output condition structures of each word without the need of a caudal neural network model, such as a GAN, and a neural network that automatically produces captions for images (Garrais et al.)

KewoNet that can recognize contextual patterns in computer screen images developed by Pivotal Kinetics. Made it possible to train a neural network using visual tasks related to speech, but also to interact with it for generating, for example, complete video text chats or video phonebills, KewoNet has been tested on a variety of language datasets, was evaluated on IMDB, Yelp and Stock price-selling data from Yahoo Finance. Pivotal Kinetics, KewoNet's new parent company, was initially founded with the goal of reimagining the world in a manner that would allow human beings to talk to machines. It was able

to launch this technology just ten years ago, but it was demonstrated as a trending machine learning solution by doing so with machine learning, not language translation[at this time this link speaks highly of]. It could be used very effectively in commercial phone applications, using intangibles to manage call-time and subscription terms, spreadsheets to store macros and product descriptions and synch text with sound. To fully understand the concept, "We are all familiar with the use of intangibles and spreadsheets as key programmatic components of corporate software, but to fully exploit phone applications, using technographic and interactive intangibles would be leveraged as a valuable asset. The possibilities of interaction between software and hardware, between supervised and unsupervised deep learning systems, between analytics agents and production executives etc. are endless, but the opportunity to totally depend on these processes must be well considered by those corporations developing their international distribution.

[Notes: What if you trained on random sets of samples taken from the train set? What if you trained on a dry version of the sounds in the training set and a wet version of them? Could it somehow whittle down to the point where it only tunes a testable level of motion or damping, it doesn't do this to an unsupervised process]

What other new techniques can we hope to see in our next blog post? To recap, we've seen here that in response to the need for

-- a loss function
-- a training set
-- the presence of the signal
-- the lack of the right audio signal/ unit characteristics
-- and the presence of human voice input

You can tune a machine to produce sounds on request which reflect these characteristics, or at least have varying amounts of them or have different qualities in different orders/ mixes. We can explore all other viable audio signals. By generating an audio product using a generative network, and then instructing a machine learning algorithm with the same set of inputs to classify each sample according to the directions the network was trained on, we can then monitor and adjust the weighting of each variable, whether the audio signal unit it is or is related to or not.

NSynth reference:
https://magenta.tensorflow.org/datasets/nsynth
Future work:

-- train AI as signal-independent parameters, and also feed them synthesized audio input

-- maybe a simple beat/harmony distinction and an input to the same trained system for each signal-independent signal
-- train with instrumental function codes
-- write a real signal generator that takes a representation of an instrument and the waveform position for its constituent signals, tries to make sense of the output waveform given a waveform, and takes that information to try to make sense of where in a sample the original waveform should be
-- conclude with a tune interpolator that tries to reproduce two given tunes

for acoustic of referent which can be either recorded or conceptualized by human-auditors,

-- classically, a mix of tone and instrumentation
-- first-to-last transition (categorization-wise) often require some ad-hoc tuning, maybe like a bass-line and melody
-- Classical, Orchestra Context, Sonic Posteriorization https://www.sonicposterisation.com/shop/context_archive come to mind for orchestral, but also other capacities and contexts that can work in conjunction with even small externalities

New work:

-- create some audio data: computer-generated, human-generated, classical form, what have you
-- take a spectrogram and a beat/melody/rhythm, etc. demarcator
-- train an autoencoder on that data
-- try to overfit to certain higher-level patterns
-- then see if it can create context or tell apart samples that previously resembled one other
-- where do you start with orchestra? I think it's mostly a matter of importing the beat/melody, so much of my early jazz stuff could find similarity there
-- from the audio dataset alone the voice characteristics would already be represented by the words, so I could compare multiple proposals for words to replicate the same internal structure
--

Previous work related to melody-smoothing: https://github.com/DeepLearning4j/dl4j-examples/blob/master/tutorial/src/main/java/org/deeplearning4j/examples/feedforward/regression/audio/AudioFileVocoder.java

It would be a different kind of loss-function if these were relatively isolated-numbers on some conversion

chart, but this isn't necessarily what we want. To get the structure of polyphony anyway I need to know the differentiators; maybe it would be a kind of feature mapping? Like an autoencoder that optimizes its output to my desired characteristic rather than anything else?

 Just how exactly do Decorator tracks (and maybe also Instrumental Accompaniment) work out with all that? All the music I've listened to several hundred times (see link in bio) and that I've recorded actually varies between a thousand and one thousand four hundred times an hour. But yeah, I'd imagine only a tiny little bit of the population actually does change their music in response to the music. (It is possible that some listeners do have the aptitude to vary frequency each time they listen) So none of those end up too far off from what the listener is used to. Mainly for Singer-Dancer and DJ tracks anyway. But I'd assume none of the listeners that exhibit actual states of consciousness vary much (based on https://www.emsltd.co.uk/kurtosis/ auditory responses to dance-music have been calibrated to have peak frequencies of 300kHz,800kHz and 1400kHz);

Train an AI to make generative sounds (20 classes: Electric Piano, Bass Guitar, Violin, Flute, Acoustic Guitar, other things, and Robot).

I have always been fascinated by audio, in particular the expressive way audio is captured and consumed.

This challenge by Jaumo is my first step into building an expressive waveform generator, based on real instruments.

My set-up

1. Audacity is my audio editor of choice. I'm using version `2.3.3-beta` running on Mac OS X 10.12.4.
2. My sample sounds are from SampleSwap.org/pop sounds.
3. My Audacity files are hosted on SoundCloud.
4... My Python libraries are `TensorFlow 1.4.0`, `numpy 1.3.3`, and `Librosa 0.5.1`.
5. Tomorrow, I want to try converting my Audacity recordings into multiple instances of GANs-trained-for-time using a few different techniques, making sure they train and produce samples of the correct length.

The next iteration of this project could be:

1. Tune and Fourier Transform to sound characteristics in the human voice
2. Build Generative Adversarial Neural Network which synthesizes audio signals: replay those using some computer model
3. Quantify and retrain with each set's harmonic variation in the form of

its harmonic correlation structure -- what's measured and modeled as loss functions or metrics that reward or penalize it
4. Get somebody with a harmonica to measure how ADTL processes it: does it make you sound like a leper?

*Note: My audacity files are pasted http://soundcloud.com/deanlab/audacity-august62019

Previous implementations:

[My version of Google's Tensorflow Audio generator](https://github.com/pincoppola/Audio_to_Geneva_Converter/blob/master/main.py), which was supposed to train with progressively higher resolution spectograms(s) → could produce any sample length and sound frequency.

My version of Google's Tensorflow Audio generator (which also took weighted average of high/low frequency audio and spectral similarity/damping factors), which was supposed to build a noise-dependent version of Adversarial Nets that streamed from the sampled audio inputs in the audio-torso(s) → raw spectogram to output audio, but absolutely with noise, during training → [here](https://github.com/pincoppola/SoundBlackSwan/blob/

master/main.py)

My version of Google's Tensorflow Audio generator (which also took weighted average of high/low frequency audio and spectral similarity/damping factors), which was supposed to train with noise dependent input(s) → based on the softwares written by Armin Braunstein at https://portfolioarmin.wordpress.com/audio-superresolution. In my model, then, the sampled audio would not be farther apart than the smoothing factor. → [here](https://github.com/pincoppola/SoundBlackSwan/blob/master/main.py)

Relevant resources:

- https://www.tensorflow.org/neural_structured_learning/overview
- https://soundcloud.com/djtoad-kidd
- http://www.audiomaze.com/Drums.pdf - essentially the article(s) from the link that state that the magnitude of frequencies characterizing rhythmic sounds is always between 500 and 1500Hz; and of course, there's the Fourier Descriptor, which maps out frequency variations at the limit between construct and decay.

chuchu train an algorithm to model musical instruments: learn how to use OpenLearner (levenshtein distance from piano, harmonica, guitar, e-bass, single string, drum rolls) and keep an eye out for gradation
foo.wav
1. Audio signal SDNN for image interpolation
2. Noise vectors using Synthetic Images
3. Use both as input/contextual cues to produce/generate/convert-from/ sample noise vectors w/o context https://arxiv.org/pdf/1509.04511v1.pdf
4. Then train a small one-dot classifier with the resulting images directly
5. If that fails, train a dot decodifier (learn top horizontal detection) with the resulting images, using systematic probing. Get image features back that correspond to the dot detector. (still probably will fail)

[Try Neural Discriminator](https://github.com/deepmind/neural-discriminator/) for optimizing for "smart projections" & write code for `attention.js`
(somewhere along the way use vectorizing w/ Pytorch)
##
note:
This used to be:
Hopelessly Dismal Adversarial

Nets for Speech Linguistics
Up until 02 September 2029
Now it's:
Adversarially Disruptive Neural Nets for Speech Linguistics & Acoustic Anomaly Detection
2020:
Up until 30 Novemberth 2020

Implementation of a basic neural network (CNN) with WGAN-GP (actually using PyTorch).

The 'actual' algorithm

Step 1. Train a classifier `Classifier` using original spectograms (not artificially sound-choked). 'Uses both GAN-textured filterbank and Spectrogram

Step 2. Identify components of a masked spectrogram that a *Classifier* assigns to a specific class that is similar to the original visual presentation of that particular sound sequence. This is accomplished with two techniques. The first one uses Regularized `Decoders`, models that assign probabilities to each possible class. It then discriminators use Regularized `Decoderers`. It then performs regularized auditory analysis (using its trained classifier as if it were a classifier) to extract neural representations, models that are trained on *image* (yaw and local) features. This is followed by decoders that assign probabilities over classes

calculated with both *`Constrainer`* and *`Decoder`*. These are compared against some benchmark model that is described in section 5.1. In the second technique, which is demonstrated below, the model assigns probabilities to be similar and is trained on different adversarial loss functions for different sound classes and then applies them to unfamiliar variated adversarial input.
The first technique using shallow 'counterfactual learning' combines different loss functions such as SOFTmax like 'Hard' and 'Harden-Dreemph' for different possible initial score vectors and normalising those vectors for a ground truth label vs. what the model will provide through that probabilistic loss function, or alternatively ''`R`''. Then it learns a decoder to attempt to assign specific pitch and harmonic frequencies length to those posteriorized masks. There is also `Unidirectional` technique described below that works similarly to the other techniques except that it trains an *`Decoder`* to extract *left* sound representations; it then trains another *`Decoder`* to extract *right* sound representations. It then assumes that it has found the right sound representations by applying regularization to a mix of those left and right sound representations by its author in a `Constraint`-inspired

```
        # still needs a voice recognition algorithm (if possible)
```

Part 4. Pretrain unsupervised acoustic descriptors of emotion (e.g.

`emotion_maps` from our previous method() — also note https://olympus-biomed.github.io/autoencoding-robustness/udivs-1002/publications/) for a similar method with font) and see if you can softly convert? UST-based solution? https://github.com/ai/rubik/blob/master/lib/soft/soft.py (douglas proposed integration of MNL-style autoencoders)

Not exactly successful, but could still train a WGAN-GP classifier on top of a MIDI player with the 'harmonics' of a piano and a bass-specific instrument (spectrograms and musical symbol ANE? https://en.wikipedia.org/wiki/MusicXML https://www.researchgate.net/publication/221783040_Extracting_Human_Distinctive_Features_from_Audio_Signals_and_Visually_Converted_to_MIDI_in_Linux_Desktop_System) and then generate video to train on.
Somewhat related: https://opensoundcontrol.org/specification-1.0 ,https://github.com/CNMAT/OSC,https://en.wikipedia.org/wiki/YouTube_Music

probably doomed

Interesting music-related code: https://github.com/vilmibm/improversive-gru-model . . . could some version of the loss function be part of music synthesis? and computational physics?

E-mailed Brad on July 5 and he replied with [this](https://www.scopus.com/inward/record.uri?eid=2-s2.0-85047984066&partnerID=MN8TOARS) . . and when I googled the reference, his co-author and I both posted on it recently . . .

Hopelessly Dismal Adversarial Nets for Speech Linguistics (Generalization Edition)

Abstract
===

A mismatch between neural representation and outputs of human speech was previously investigated in a computer-aided visual search web application (I know this because it says so on the conference poster in question), which proved that human speech could train an algorithm to recognize and produce distinctive representations. While implementing this approach is not computationally expensive in a large dataset, no such network that is able to generate the whole language space was described. In this must-try-it-harder recipe we suggest a novel way to generate speech features, whose quality is of critical importance to a vocoder and autoencoder combination of neural networks. The major idea behind this recipe is to identify the acoustic characteristics of a number of speech samples and to try to generate with them a phoneme-autoencoder. We posit that the most popular form of speech autoencoder is trained with

variations of short repetitive, doubly sine-pitched vectors and that vocoders are designed to adequately capture latent representations.

We are going to learn this by generously providing the authors with some local images and sound layers to extract part of their representation that is proportional to only 28 pixels (28 x 1 pixels). Then we will apply labels to the resulting data and then use a generative autoencoder to predict the value of the implicit hypothesis. We train the inference network of the linguistic representation using human session speaker type (Dan?)acoustic features and multiple variables (we also hope that it works better with confidence labels as in the paper). The generative model, which will resemble the cycle-gan[cite me] model or DWTNet [use training with mini-batches for stability], will be trained on the target speaker type
For example, if the authors hold each image or acoustic vector for the associative representation of the speaker class for 100 epochs, we expect a relatively low 'loss' (typical 1000 epoch runs 0.005-0.003) such that the 16 x 10-Conv-Window Convolutional Kernel is approximately 3/2 smaller than the real size. We will confirm this by creating an arbitrary model based on the same raw acoustic representation. The model classifier loss will then be comprised of three convolutional layers, a softmax layer and an MSE loss that measures the

similarity between the imagined speech and the real audio. Our final autoencoder networks will contain 5-6 convolutional layers and 5 full-connected layers with a softmax output and some pre-trained weights.

More background information and motivation

Building a decent synth
An [attempt to synthesize sounds](https://github.com/tensorflow/tensor2tensor/issues/348) with iterative and multiple Delauney extrapolation . . . [not much of a success](https://github.com/tensorflow/tensor2tensor/issues/381). . . . . [Idea](https://github.com/osiebler/wavenet_vocoder/issues/57) for the fact that the Spectrogram made it quite possible for different samples to sound the same and should probably be cleaned up. It seems to me without this, this is not even a top priority and would only be interesting if one started with a certain feature, which therefore allowed itself to come up again in a new feature set. This is especially if the underlying idea of a series of oscillations as a 'test' is too weird to be proper otherwise. However, the best case scenario (which I'm quite sure to occur), a few features are removed and the other features are added so that the spectral representation of the more 'true representation' of the input actual sound is retained. (In version

0.9.2 [Deprecated: /tmp/setup.py|0|13.10.7] of that link, `/tmp/setup.py` will change . . .) I want to try some sort of 'zig-zagged' iteration of the feature space[as adescribed here][as described in the link], though maybe I'll try to add some features based on their fixed means and variances and this code would do that too. The neutral ground here between all is the only way to build 'new' ones. I certainly am uncomfortable with those ([Spectral oversampling](https://github.com/amsehili/ESPNet#spectral-oversampling-style-loss-for-mel-mixture)), and would rather use a decoder that does not contain any characters in its word vector. To be honest, as I stated above, I don't know how to actually fix this with a standard recursive algorithm, but I'm sure even that could be solved with an autoencoder. (And just wanted to say that your approach to the problem has been absolutely the most scientifically and scientifically exciting one, so I thank you for your willingness to investigate this issue)

Seeing if it works

I *did* find a source of masked audio data in [this GitHub folder](https://github.com/yu4u/noise2noise), so I used [this module](https://github.com/yu4u/noise2noise/Oe/models/recurrent/bingwa_resnet32.py) to both generate and predict some noisy samples. Then I tested those results synthetically to come up with some `pytorch` code.

The first code was essentially an LSTM sequence-to-sequence translation feature + gcd layer used by Wavenet to convert STFT representation to raw audio. Note that this link also offers a single input signal from the three domains: frequency/amplitude, timbre and quantitative metrics, but my script is mainly inclusive of the "functional-synopsis" portions of the signal for amplitude. So I removed the visual part, added some LSTM kernels and be forgotten about it.

Based on [spectrogram prediction (based on Song Hyouk)](https://github.com/songsth/week-generation), [based on Spectral masks (based on Alexander Semov)](https://github.com/semov/waves/blob/master/model/spectrogram.py) and a model query lambda propagation system based on [Iman Sbitouri FCNNs (based on Wan-Lun Tai)](https://github.com/wand140wa1) (and I'm sure it could be improved by converting some of the network's layers, particularly by resampling the convolutionally decompressed audio, possibly applying a gcd layer, then using those elevated-performance fine-tuned parameters to adapt the `Bayes Layer` layer from the A.C.-based model (these optimized layers will be fed equalized FM and the convolution output will be updated accordingly to produce the acoustic features calculated using `weavy`), obtaining [the resonant frequencies of the input

audio signal](http://www.ideaapps.asia/uploads/1/0/7/1/107174558/musicdepot305.pdf).

This resulted in a mixture (of the latter 'by analogy' with the standard syntax for signal synthesis, where the same typemorphs regular) which looks like [an LSTM](https://arxiv.org/pdf/1711.10433.pdf) (or rather an convolutional layer + an LSTM) trained on synthesized audio from various sources such as sampleMusic, Open Source Air-Lines, and music from audiobooks

Am I wrong that the resulting audio is then converted to the target data? This is basically the idea for speckles

To do: provide the masked spectrograms in input(s) and test a number of architectures (including DCGAN, WGAN, BEGAN, and possibly vgenerative)

Other notes
I loaded using TensorFlow's [MIR-BLEAK](https://deepmind.com/blog/spectrogram/) Dataset

I also tried a few different [pretrained audios](http://tenor.sd2cd.com/infer_pretrained_mind.pdf) and found them mostly insufficient. Of particular interest, the Simple Audio `jorine` is good but the generated spectrograms and sounds vary quite a bit

The '[Random Sampling Playgrounds]

(#https://guz-vol/random-sampling-playgrounds)' volume contained most . . . . . samples on audio datasets.

The '[musicnet](#https://github.com/fivelfliss/musicnet) repository contained lots of noise samples.

(s) Some sound output for visual analysis

* (s) First, the output from the Soft Gumble sampler from [this concept](#http://gradientscience.org/blog/hidden-structure-internal-representations-are-essential), using a "linear model" as a basis and using it in place of real-world audio. The gradation is just a fun trick to get some amazing [convolutional stereo](http://www.oozenet.net/cnconv/) images[this atraditionally called 'unnatural']. :)
* (