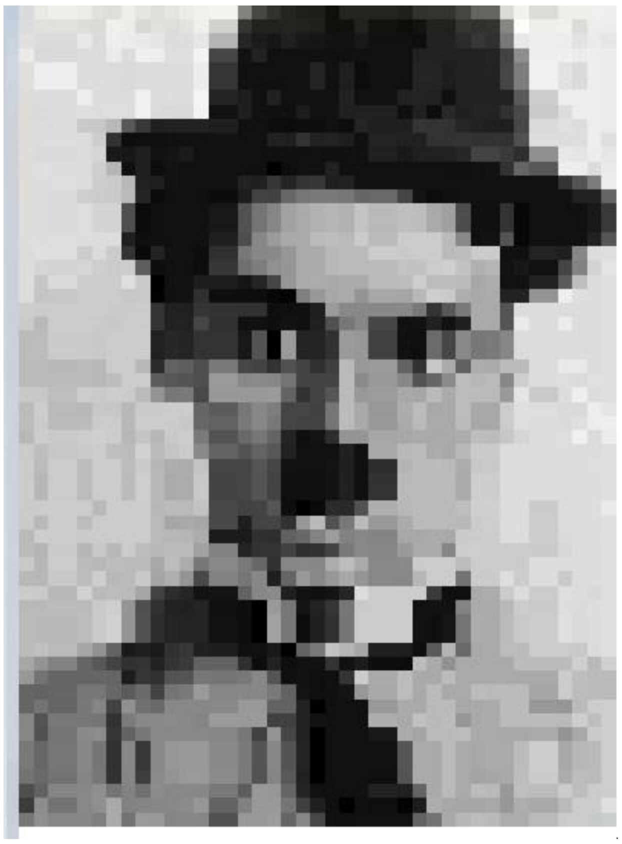
# An Introductory Guide to Understand how ANNs Conceptualize New Ideas (using Embedding)

**Introduction**

Here's something you don't hear everyday – everything we perceive is just a best case probabilistic prediction by our brain, based on our past encounters and knowledge gained through other mediums. This might sound extremely counter intuitive because we have always imagined that our brain mostly gives us deterministic answers.

We'll do a small experiment to showcase this logic. Take a look at the below image:



Q1. Do you see a human ?

Q2. Can you identify the person?

Q3. Can you identify the color of the clothes this person is wearing?

Keep these questions in mind. We will get back to them after seeing what we will be covering in this article.

*Understanding Artificial Neural Networks becomes easy if you draw parallels with the working of the human brain. Today, we will explore the concept of embedding. We will first understand what "embedding" means in our human brain space, and then look at it's applications and use cases.*

**Table of Contents**

1. A simple thought experiment

2. Embedding in the human brain
3. Embedding in Artificial Neural Networks
4. Applications of semantic embedding
5. Learning embedding from data
6. Formulation of text embedding
7. Popular word embedding algorithms
8. Image and speech embedding
9. One shot learning
10. Siamese Network

## A simple thought experiment

Let’s get back to our experiment. Recall the blurred image and our three questions. All of you must have guessed the answer to the first question, most of you must have guessed the second one, and a few of you might have guessed the third one. Give a score on your confidence level (out of 100) for each of the three questions and average them. Why do you think you did not score an absolute 100? The reason is simple, our brain does not have enough data to identify this image with a probability ~ 1.0 . Now look at the image below:

Try to answer the three questions again. Did you get a 100 this time? You might think that our brain is returning a deterministic value for the three questions this time. But it turns out that your brain is making predictions that are very, very close to 1.0 probability because this time our brain got enough information.
Let’s make this experiment more interesting. Look at the first image again and compute your score. Did you score higher than before? If yes, the reason is simple. Your brain has used its past memory to add new information which is not even there in the picture and increased the returned confidence value. This new information is that the brain has seen an almost similar picture before.
The above experiment demonstrates that our brain tries to predict everything around us and works with the best guess. This statement is true to such an extent that our brain even has to predict the location of parts of our own body through experience. The brain cannot even be deterministic of the facts like where your hand or legs or chest are located. Google “body transfer illusion” to read about experiments that will prove this fact.

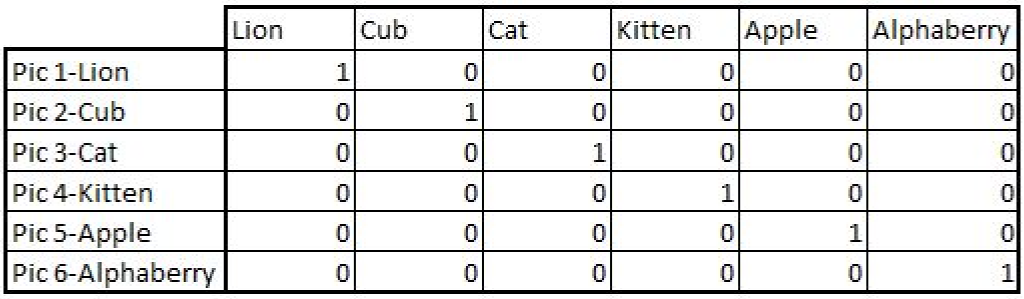
## Embedding in the human brain

Coming to the big question – if our brain predicts everything with past experience or data, how do we work so well in situations where we have no prior experience or knowledge? For instance, if you go to a grocery store and get a new fruit called “Alphaberry”, what will you do with this item?
Probably keep it in the refrigerator, wash it and eat it. How do you know so much about what to do with this new item even when you have no experience with the subject? It turns out that our brain tries to create a semantic understanding of everything and refines this understanding with experience. This initial semantic understanding gives it a head start.

For instance, the brain knows that “Alphaberry” is a fruit, hence it will share a lot of property with other fruits, i.e., we can eat it without any danger. Once we have eaten this fruit, our brain adds the new information to refine the semantic representation of “Alphaberry”.
These semantic representations can be used by the brain to find similarities between concept/objects or draw analogies or make reasoning. The name “Alphaberry” is just the address of this semantic representation in our memory. The moment I ask you – “Have you tried alphaberry?”, your brain will look for semantic representations of “alphaberry” and retrieve all the experiences/information about the fruit. It then evaluates probabilistic answers to the question and gives you the appropriate response – “Yes. It was very sweet.”
Let’s take another example. What is 20 * 10 ? Obviously 200. Numbers, unlike words and pictures, have their semantics encoded in itself. Hence, our brain does not see 10 and 20 as addresses of semantics but as semantics itself. Our brain can directly do operations on these numbers unlike “alphaberry”, where it had to retrieve semantic representations before it could answer anything about the subject. This is an important concept and we will refer this concept later in this article.
The above description, as you can imagine, is an over simplification of what really goes in our complex brain. In the next section, we will talk about a concept used in Artificial Neural Networks (ANN), that is parallel to semantic representation in humans.

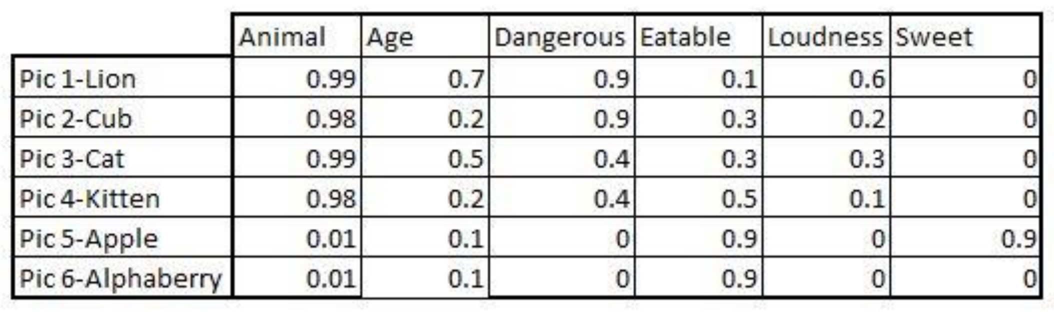
## Embedding in Artificial Neural Networks (ANN)

In the last decade, we have made computers highly efficient with numbers (even better than humans). Computers can only work with numbers as they are the only entities that have semantic encoded on itself. How can we make computers understand concept like words, images, audio or videos? The answer lies in our previous section.
We need a “semantic representation” of all these concepts in numbers or array of numbers, as numbers are all computers can understand. It is important that the array of numbers do a good job of denoting the semantics of the entity, else we will have a “garbage in garbage out” kind of model.
We will contrast two models to denote the semantics in order to get a deeper understanding of this concept. Here we will talk about representation of 6 concepts – “Lion”, “Cub”, “Cat”,”Kitten”,”Apple”, “Alphaberry”. You have 6 pictures of each and need to represent them in numeric form.
Our first model is one-hot encoding vector representation of words. The representation looks as follows:



| | Lion | Cub | Cat | Kitten | Apple | Alphaberry |
|---|---|---|---|---|---|---|
| Pic 1-Lion | 1 | 0 | 0 | 0 | 0 | 0 |
| Pic 2-Cub | 0 | 1 | 0 | 0 | 0 | 0 |
| Pic 3-Cat | 0 | 0 | 1 | 0 | 0 | 0 |
| Pic 4-Kitten | 0 | 0 | 0 | 1 | 0 | 0 |
| Pic 5-Apple | 0 | 0 | 0 | 0 | 1 | 0 |
| Pic 6-Alphaberry | 0 | 0 | 0 | 0 | 0 | 1 |

Our second model is based on attributes of each picture. This is more like the semantic/meaning of the picture. Following is an example:



| | Animal | Age | Dangerous | Eatable | Loudness | Sweet |
|---|---|---|---|---|---|---|
| Pic 1-Lion | 0.99 | 0.7 | 0.9 | 0.1 | 0.6 | 0 |
| Pic 2-Cub | 0.98 | 0.2 | 0.9 | 0.3 | 0.2 | 0 |
| Pic 3-Cat | 0.99 | 0.5 | 0.4 | 0.3 | 0.3 | 0 |
| Pic 4-Kitten | 0.98 | 0.2 | 0.4 | 0.5 | 0.1 | 0 |
| Pic 5-Apple | 0.01 | 0.1 | 0 | 0.9 | 0 | 0.9 |
| Pic 6-Alphaberry | 0.01 | 0.1 | 0 | 0.9 | 0 | 0 |

Now let's try to contrast. Try answering the following questions with both the representations individually.

1. If you have no idea what "Alphaberry" is, which of the two approaches gives at least some basic understanding of the item?
2. Which of the two can draw logical conclusions like "if kittens are younger cats, what do we call younger lion?"
3. If I want to eat something sweet, what can I eat out of the above six options? Imagine if our brain gave us the output "lion", obviously we never question our brain, do we?

I am sure you already know that the second representation did well on all the above 3 questions. Not only does the semantic representation give us a head start on new concepts like "Alphaberry", it also helps us infer logically. For instance if you do the following task mathematically, you will find the answer to our second question :

```
Vector (Pic 1-Lion) + Vector(Pic4-kitten) - Vector(Pic3-Cat) = Vector(Pic2-Cub)
```

How awesome is that! Now we can do mathematical operations on abstract concepts like words/pictures/audio. We also know semantic representation in form of numeric vectors for abstract concepts can be very helpful for our ANN models. But how can we create so many semantic representations for all the words/images/audios in this world? The answer is simple – by learning from lots and lots of data. The features from such a learning will be a lot more abstract unlike the one we used in our demonstrative example above.

## Applications of semantics embedding

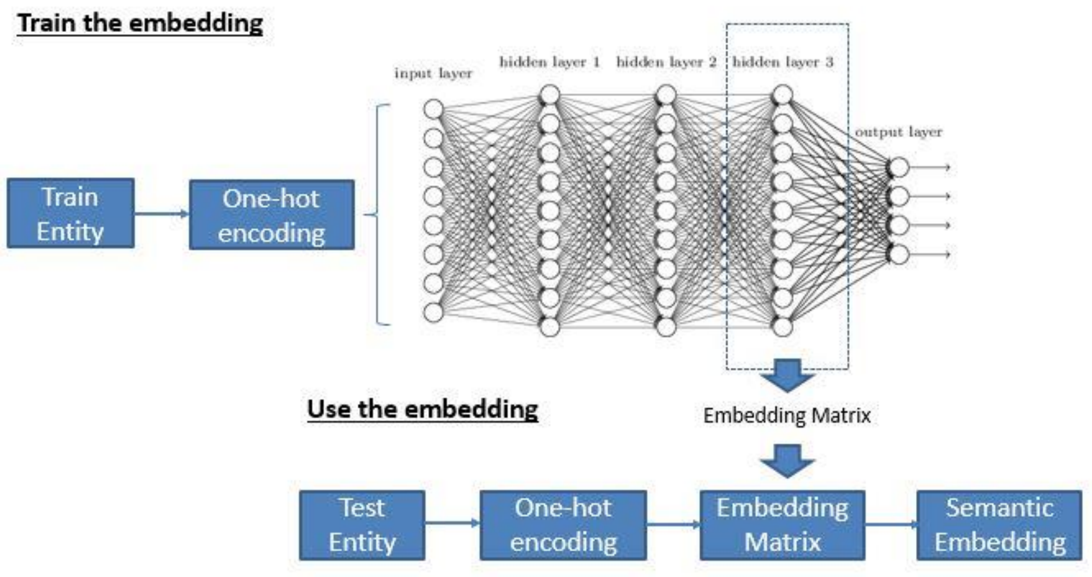
Like our brain uses semantics in all the cognitive tasks, Artificial Neural Networks use semantic embedding for numerous tasks. We will categorize these applications under 3 main types of embedding they use.

1. **Word Embedding** – Word embeddings are used everywhere in natural language processing (NLP). A few important examples are sentiment analysis, topic modelling, etc.
2. **Image Embedding** – Image embedding is another important space of active research. Important applications include face authentication, face identification, etc. For instance, Baidu uses image embedding to match an image of a person entering their office premise with the employee data base. Have a look at this research paper if you are interested in reading more about this application in more detail.
3. **Speech Embedding** – This one saves a lot of money for call centers. One of the strongest applications of speech embedding is voice biometric. Using this, call centers capture voice print (very similar to finger prints), and identify & authenticate customers.

Let us now try to understand a generic concept of learning embedding.

### Learning Embedding from data

The underlying methodology for developing any type of embedding is almost the same. We prepare a training data with one hot encoded entity (this can be a word or image or audio), define some kind of target function and develop a neural network. We then throw out the last layer of the neural net and use the weights of the intermediate layer as the embedding. Following is the generic process flow:

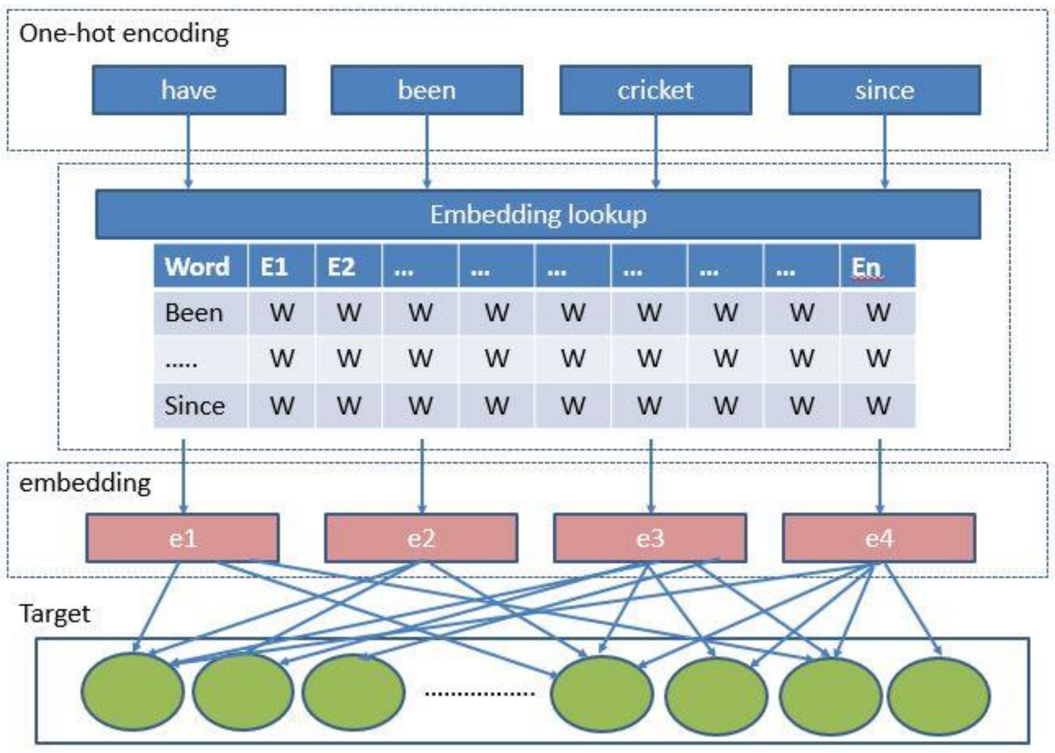


### Formulation of Text Embedding

Most of the algorithms that are used to train text embedding work on a very simple framework. Let's try to understand this with an example. We will use the following sentence to illustrate the logic:

```
I have been playing Cricket since my childhood.
```

- We will choose a target word randomly from this sentence, say – "playing"
- Now we choose a hyper-parameter "range = 2". This means we will choose context words in +/- 2 words from the word "playing"
- Our context vector becomes – ["have","been","cricket","since"]
- Create one-hot encoding vectors for all the context and target words
- Initialize embedding matrix with number of rows as vocabulary size and number of columns as number of features we need
- Now we define our  Neural Network architecture to learn our embedding matrix



| Word | E1 | E2 | ... | ... | ... | ... | ... | ... | En |
|---|---|---|---|---|---|---|---|---|---|
| Been | W | W | W | W | W | W | W | W | W |
| ..... | W | W | W | W | W | W | W | W | W |
| Since | W | W | W | W | W | W | W | W | W |

- Train the network and get the final weights in the embedding matrix

**Popular word embedding algorithms**

Word2vec – Word2vec is the most popular embedding algorithm. It works on very simplistic scenarios of the generic algorithm. We randomly choose a target word and then choose one word from the generic context vector as the final context. For instance, we can choose “playing” as the target vector and “cricket” as the context. Now we run the generic model that we discussed in the last section.

The only limitation of this methodology is that the calculation of the *softmax* function at the end of the framework is extremely expensive. This is because the size of the output node is equal to the vocabulary size (which is generally more than 10k). You can always use pre-trained *word2vec* matrix for your business case to avoid this computational cost.

Negative Sampling – This is another powerful concept which gets rid of the one challenge in *word2vec*. Instead of multiple output nodes, we convert the problem into a binary classification. The target word is chosen at random, as before. For the context word, we initially choose one of the correct proximity words, and then choose random words from the dictionary. Each of these pairs are seen as separate observations.

In the above table, we use the field “correct” as the output node and treat this model as a binary classifier. Hence, we can avoid the expensive calculation of *softmax* function in the *word2vec* algorithm.

Other algorithms – Many other algorithms such as GloVe word vectors, etc. have been used in the industry. All the models work on the same generic architecture with minor changes. If you have a small dataset, it is generally recommended to use pre-trained embedding. These embeddings have been trained on millions of documents and hence have very accurate semantic information.

### Image & Speech Embedding

Even though word embedding is the most popular application of embedding, image & speech embedding is no less when it comes to practical applications. The main use of both image and speech embedding is authentication. We authenticate customers in every industry before we share any private information. There is a good chance you have encountered the use of embedding without even knowing it. Consider the following examples:

1. You unlock your phone through your fingerprint
2. You ask Google Home/Alexa to tell you your schedule for the day
3. Your Facebook account identifies all your friends automatically without a hint
4. You get verified through your voice imprint when you call your bank

Hopefully you can relate to some of the above use cases. Each of these is primarily based on image or speech embedding. Both speech and image are analyzed with the same objective, i.e., to find similarity between multiple voice/images, with almost the same architecture. The only difference is that speech is first converted to image using filter banks/MFCC in order to visualize how humans perceive sound. Then it follows the same process as the formation of image embedding. In both speech and image, we see two broad use cases:

1. Verification – With a known pair of images or sounds, we try to validate the similarities. For instance, when you unlock your phone using your fingerprint, the phone knows the owner and just tries to do a one-to-one match of the image. This application is called verification and is a computationally less expensive use case. Another similar use case in voice will be asking Google Home for any personal information. Google Home will first try to verify it before giving you this private information.
2. Recognition – This one is a much more complex task. Here, the algorithm is trying to identify the person from a number of possibilities. For instance, when you post a picture on Facebook with a friend, Facebook's algorithm tries to match the face of your friend with all your friends. If it finds a match, it will make the suggestion. Recognition is even more complex if we are using it for both identification and verification. Baidu has implemented such a system to identify employees at the entrance gate. The reason it is more complex is that the algorithm has to make enormous number of matches and then return TRUE only if it is extremely confident that it is a match. For Baidu's system, it also distinguishes whether the face is for a live human, or just a static picture. This feature makes the facial recognition system extremely useful.

### One shot learning

Why do we need embedding for the verification/recognition tasks? Why can't we train a model for each face/voice separately? We already know neural networks need a lot of data to be precise and accurate. However, most of our verification/recognition tasks are one shot learning. One shot learning is learning made on one, or very few examples. For instance, Baidu's system will probably have 1 or 2 pictures of every employee. How can we create a model when we have just a few data-points per class? This is the reason we create embedding for each image and then try to find similarities between the embeddings. This concept will become clearer once we are done with the neural network architecture of training image/voice embedding.

## Siamese Network

The neural network architecture for training image embedding is commonly known as Siamese Network. I have included only one out of the many algorithms that are used to create image embedding. In this method, we randomly choose two images from our population and send them through shared CNN stacked layers. The vector we get as output is the image embedding. Then we take a distance/difference between the two embeddings. This difference is finally passed through an activation function to check if the image is of the same person.
Note that the embedding matrix we get in this process is not for any unique person, but to find features that can tell us “How similar the two faces look”.

## End Notes

I hope this article has given you a strong foundation in the concept of embedding and helped you understand how important it is when it comes to analyzing unstructured data. In simple terms, we are trying to create a structured data out of unstructured underlying data using these embeddings. This structured data has the meaning of underlying data embedded in form of a vector and hence the name “embedding”.