
Sefik Ilkin Serengil




Developer's Log

Menu ▾

Autoencoder: Neural Networks For Unsupervised Learning

March 21, 2018 / Machine Learning

Ad ▾




View
Images

Make
money

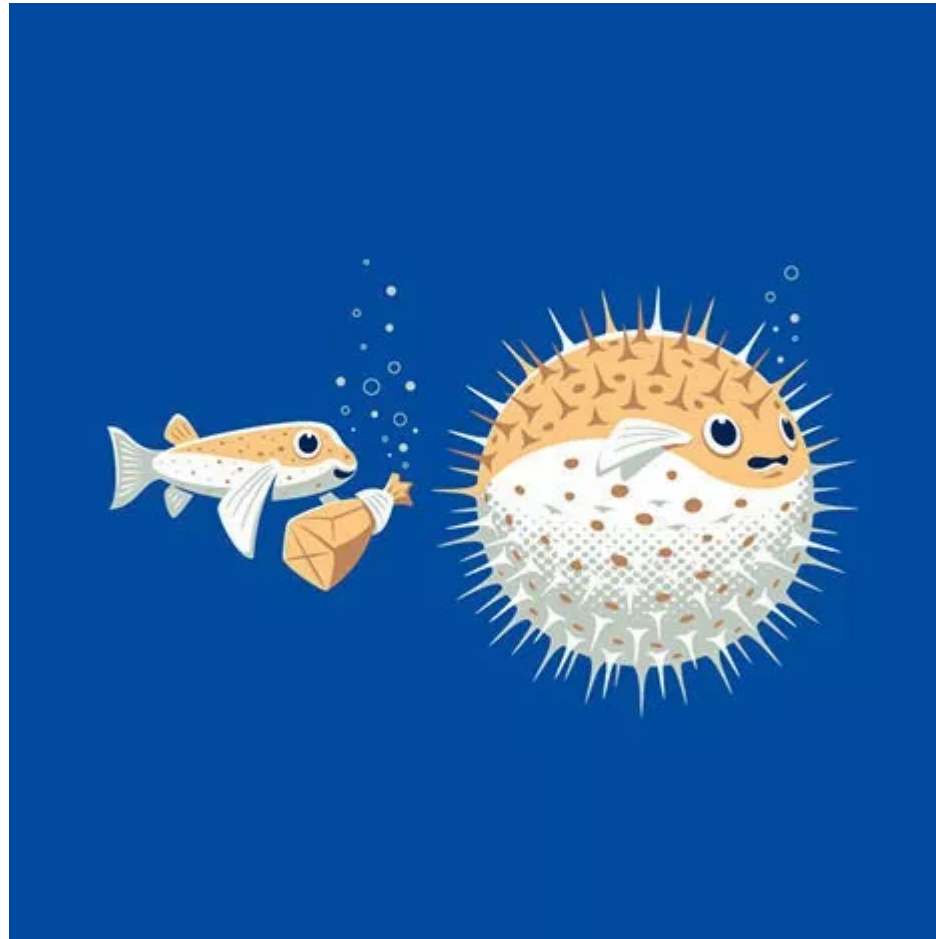
Earn more
than

\$50 /day



Earn

Neural networks are like swiss army knives. They can solve both classification and regression problems. Surprisingly, they can also contribute unsupervised learning problems. Today, we are going to mention autoencoders which adapt neural networks into unsupervised learning.



Blowfish as compressed and uncompressed

They are actually traditional neural networks. Their design make them special. Firstly, they must have same number of nodes for both input and output layers. Secondly, hidden layers must be symmetric about center. Thirdly, number of nodes for hidden layers must decrease from left to centroid, and must increase from centroid to right.



Jazz

**KRAZY
KARACH
OFFER**

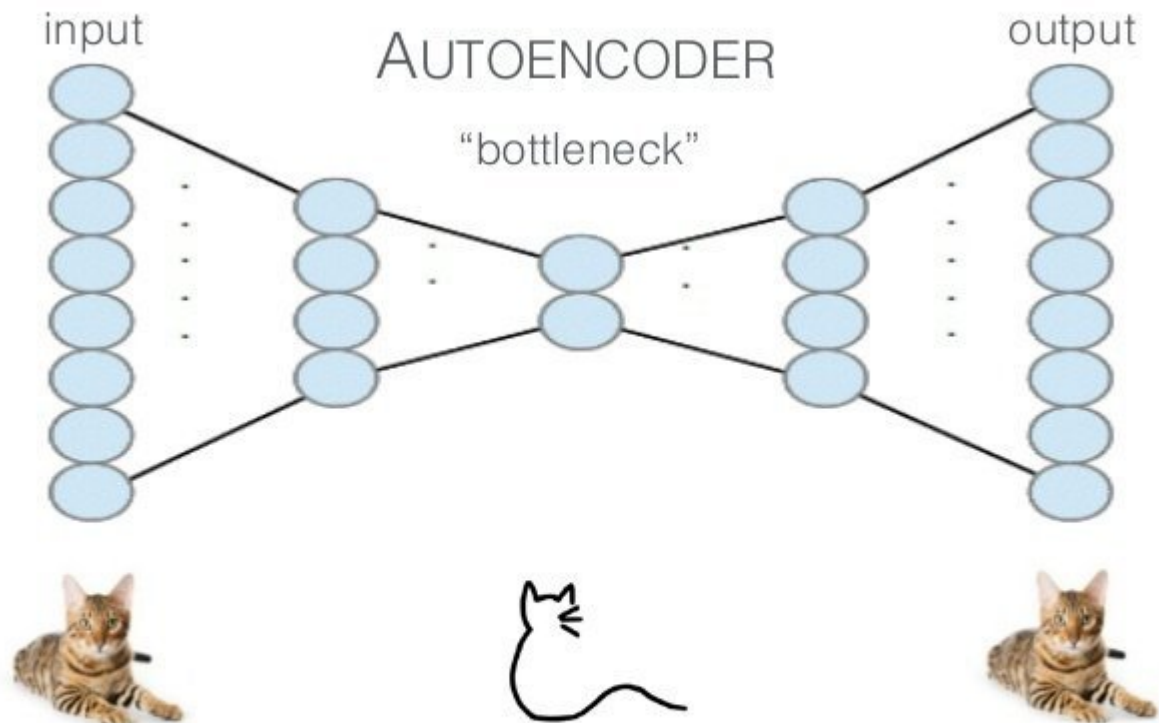
5000 | **5000**
MB & SMS | Jazz World Min
FOR 7 DAYS | 08

For Rs. 175

Dial ***406#**

[Click Here](#)

dunya ko bataa do



Autoencoder Neural Networks

The key point is that input features are reduced and restored respectively. We can say that input can be compressed as the value of centroid layer's output if input is similar to output. I said similar because this compression operation is not lossless compression.

Left side of this network is called as autoencoder and it is responsible for reduction. On the other hand, right side of the network is called as autodecoder and this is in charge of enlargement.

Let's apply this approach to handwritten digit dataset. We've already applied [several approaches](#) for this problem before. Even though both training and testing sets are already labeled from 0 to 9, we will discard their labels and pretend not to know what they are.

Constructing network

Let's construct the autoencoder structure first. As you might remember, dataset consists of 28×28 pixel images. This means that input features are size of 784 (28×28).

```
1 model = Sequential()
2 model.add(Dense(128, activation='relu', input_shape=(784,)))
3 model.add(Dense(32, activation='relu'))
4 model.add(Dense(128, activation='relu'))
5 model.add(Dense(784, activation='sigmoid'))
```

Autoencoder model would have 784 nodes in both input and output layers. What's more, there are 3 hidden layers size of 128, 32 and 128 respectively. Based on the autoencoder construction rule, it is symmetric about the centroid and centroid layer consists of 32 nodes.

Training

We'll transfer input features of trainset for both input layer and output layer.

```
1 model.compile(loss='binary_crossentropy', optimizer='adam')
2 model.fit(x_train, x_train, epochs=3, validation_data=(x_test, x_test))
```

Both train error and validation error satisfies me (loss: 0.0881 – val_loss: 0.0867). But it would be concrete when it is applied for a real example.

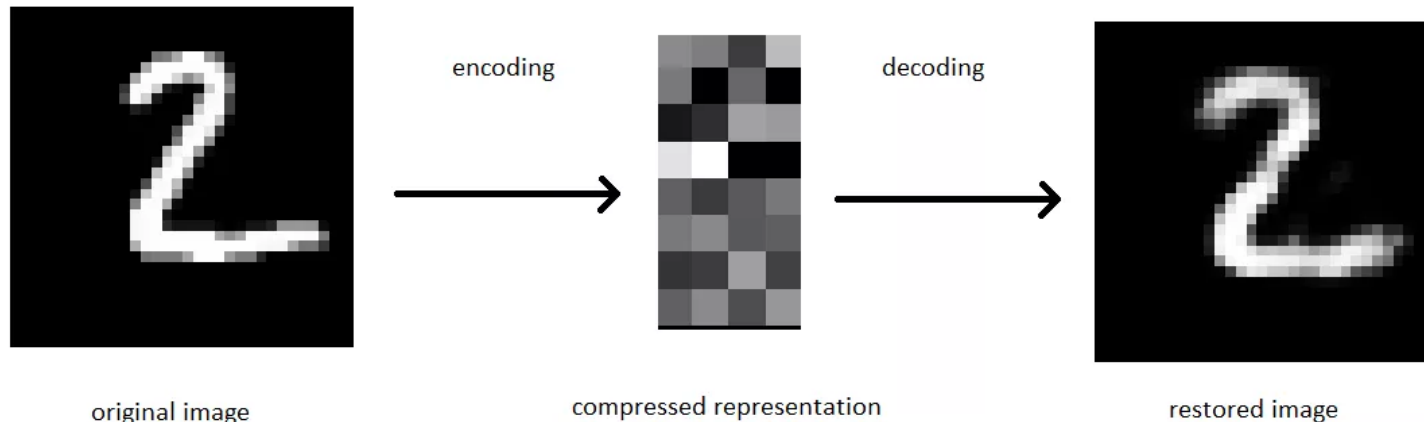
Testing

```
1 def test_restoration(model):
2     decoded_imgs = model.predict(x_test)
3     get_3rd_layer_output = K.function([model.layers[0].input], [model.layers[1].output])
```

```

4
5 for i in range(2):
6     print("original: ")
7     plt.imshow(x_test[i].reshape(28, 28))
8     plt.show()
9     #-----
10    print("reconstructed: ")
11    plt.imshow(decoded_imgs[i].reshape(28, 28))
12    plt.show()
13    #-----
14    print("compressed: ")
15    current_compressed = get_3rd_layer_output([x_test[i:i+1]])[0][0]
16    plt.imshow(current_compressed.reshape(8, 4))
17    plt.show()

```



Running autoencoder

Even though restored one is a little blurred, it is clearly readable. Herein, it means that compressed representation is meaningful.

We do not need to display restorations anymore. We can use the following code block to store compressed versions instead of displaying.

```

1 | def autoencode(model):

```

```

2   decoded_imgs = model.predict(x_test)
3
4   get_3rd_layer_output = K.function([model.layers[0].input], [model.layers[1].output])
5   compressed = get_3rd_layer_output([x_test])
6
7   return compressed
8
9   com = autoencode(model)

```

Clustering

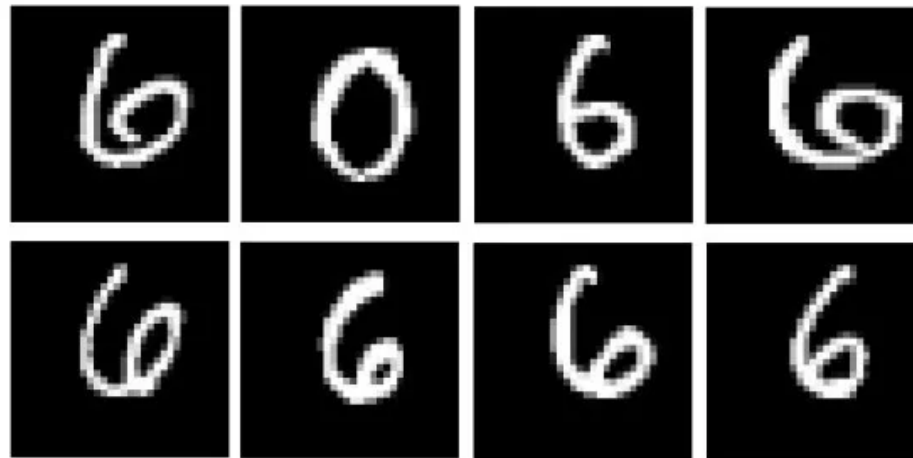
Notice that input features are size of 784 whereas compressed representation is size of 32. This means that it is 24 times smaller than the original image. Herein, complex input features enforces traditional unsupervised learning algorithms such as k-means or k-NN. On the other hand, including all features would confuse these algorithms. The idea is that you should apply autoencoder, reduce input features and extract meaningful data first. Then, you should apply a unsupervised learning algorithm to compressed representation. In this way, clustering algorithms works high performance whereas it produces more meaningful results.

```

1   unsupervised_model = tf.contrib.learn.KMeansClustering(
2       10
3       , distance_metric = clustering_ops.SQUARED_EUCLIDEAN_DISTANCE
4       , initial_clusters=tf.contrib.learn.KMeansClustering.RANDOM_INIT)
5
6   def train_input_fn():
7       data = tf.constant(com[0], tf.float32)
8       return (data, None)
9
10  unsupervised_model.fit(input_fn=train_input_fn, steps=5000)
11  clusters = unsupervised_model.predict(input_fn=train_input_fn)
12
13  index = 0
14  for i in clusters:
15      current_cluster = i['cluster_idx']
16      features = x_test[index]
17      index = index + 1

```


Surprisingly, this approach puts the following images in the same cluster. It seems that clustering is based on general shapes of digits instead of their identities.



Items of Cluster 4



Items of Cluster 1

So, we've mentioned how to adapt neural networks in unsupervised learning process. Autoencoders are trend topics of last years. They are not the alternative of supervised learning algorithms. Today, most data we have are pixel based and unlabeled. Some mechanisms such as mechanical turk provides services to label these

unlabeled data. This approach might help and fasten to label unlabeled data process. Finally, source code of this post is [pushed](#) to GitHub.

Share this:

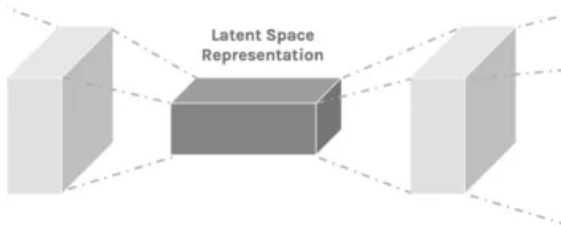


Like this:

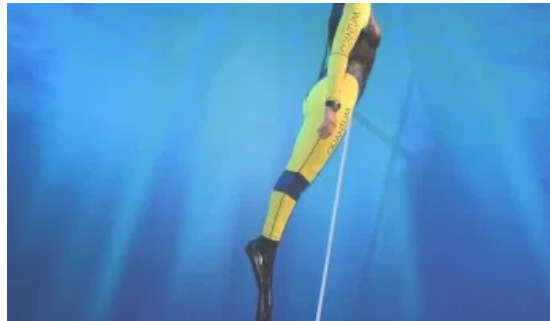
Like

Be the first to like this.

Related



Convolutional Autoencoder: Clustering Images with Neural Networks



From Neural Networks To Deep Learning



Backpropagation Implementation: Neural Networks Learning From Theory To Action

[#autoencoder](#), [#neural networks](#), [#unsupervised learning](#)

[« Previous](#) / [Next »](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

ex: jane doe

Email *

ex: janedoe@gmail.com

Website

ex: http://janedoe.wordpress.com

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

Submit

This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)

You can subscribe this blog and receive notifications for new posts

Email *

I'm not a robot

reCAPTCHA
[Privacy](#) - [Terms](#)

Follow Blog

You can use any content of this blog just to the extent that you cite or reference
