Jeremy Jordan — Variational autoencoders.   Subscribe

Variational autoencoders.

JEREMY JORDAN
19 MAR 2018 • 8 MIN READ

In my introductory post on autoencoders, I discussed various models (undercomplete, sparse, denoising, contractive) which take data as input and discover some latent state representation of that data. More specifically, our input data is converted into an *encoding vector* where each dimension represents some learned attribute about the data. The most important detail to grasp here is that our encoder network is outputting a *single value* for each encoding dimension. The decoder network then subsequently takes these values and attempts to recreate the original input.

A variational autoencoder (VAE) provides a *probabilistic* manner for describing an observation in latent space. Thus, rather than building an encoder which outputs a single value to describe each latent state attribute, we'll formulate our encoder to describe a probability distribution for each latent attribute.

Intuition

To provide an example, let's suppose we've trained an autoencoder model on a large dataset of faces with a encoding dimension of 6. An ideal autoencoder will learn descriptive attributes of faces such as skin color, whether or not the person is wearing glasses, etc. in an attempt to describe an observation in some compressed representation.
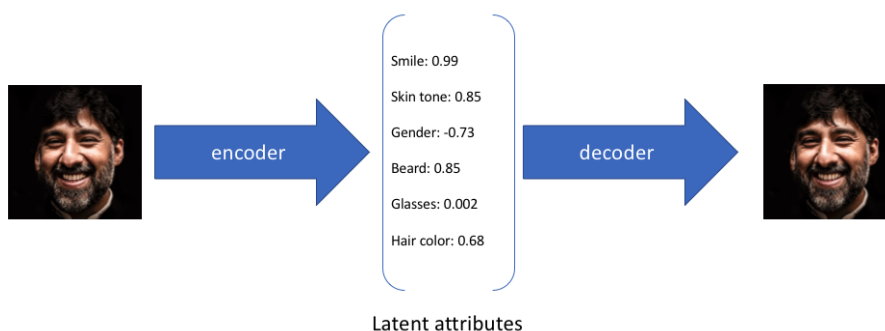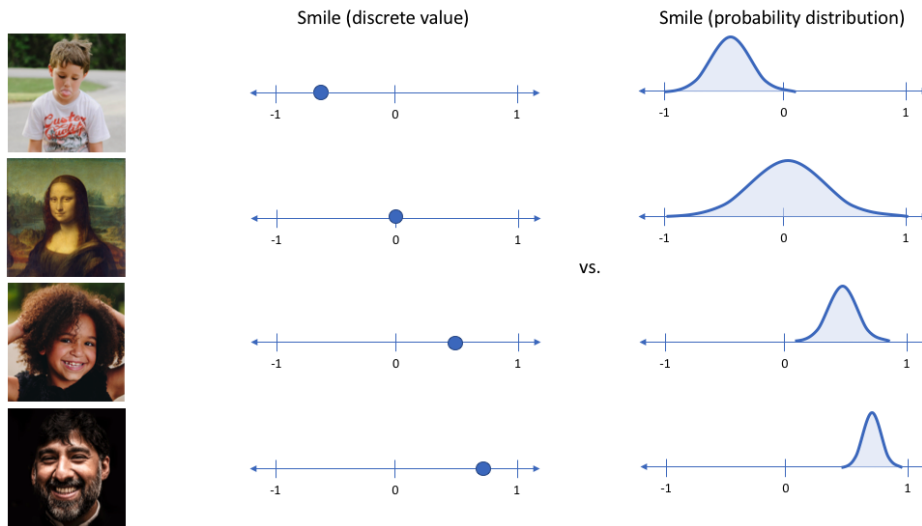

Latent attributes

In the example above, we've described the input image in terms of its latent attributes using a single value to describe each

attribute as a range of possible values. For instance, what *single value* would you assign for the smile attribute if you feed in a photo of the Mona Lisa? Using a variational autoencoder, we can describe latent attributes in probabilistic terms.

With this approach, we'll now represent *each latent attribute* for a given input as a probability distribution. When decoding from the latent state, we'll randomly sample from each latent state distribution to generate a vector as input for our decoder model.
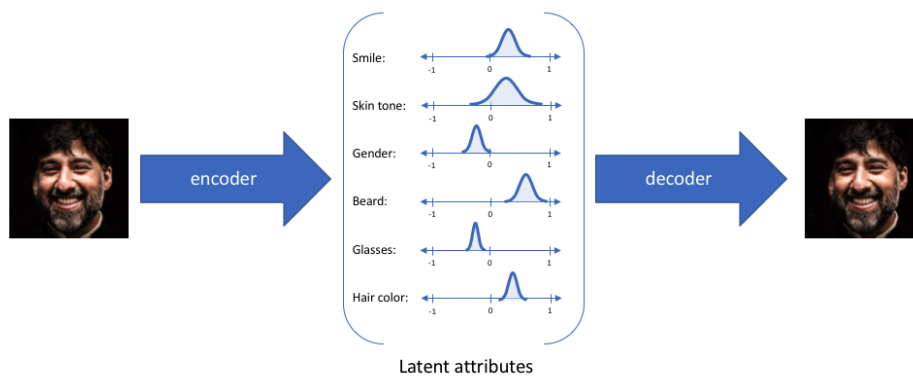
*Note: For variational autoencoders, the encoder model is sometimes referred to as the **recognition model** whereas the decoder model is sometimes referred to as the **generative model**.*

By constructing our encoder model to output a range of possible

enforcing a continuous, smooth latent space representation. For

decoder model to be able to accurately reconstruct the input. Thus, values which are nearby to one another in latent space should correspond with very similar reconstructions.

Latent distributions   Sampled latent attributes

We expect an accurate reconstruction for any sample from the latent state distributions
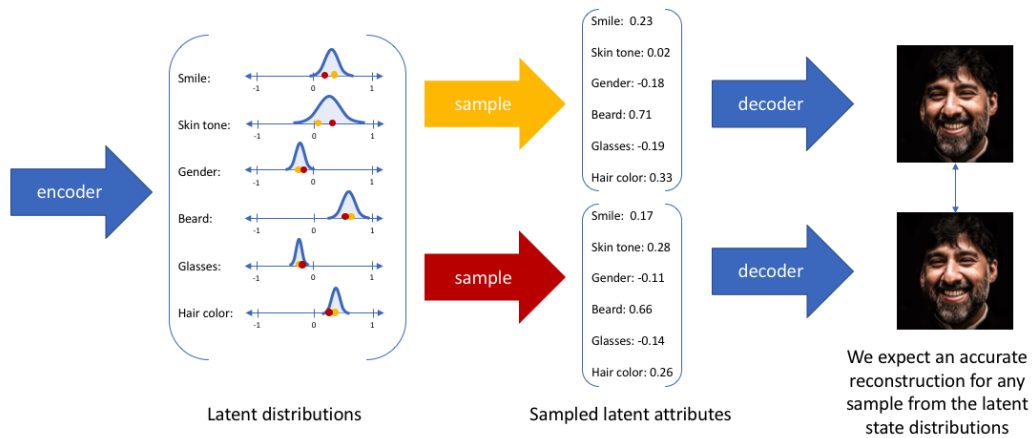
Statisical motivation

Suppose that there exists some hidden variable z which generates an observation x.

We can only see x, but we would like to infer the characteristics of z. In other words, we'd like to compute $p\left(z|x\right)$.

$$p\left(z|x\right) = \frac{p\left(x|z\right)p\left(z\right)}{p\left(x\right)}$$

Unfortunately, computing $p\left(x\right)$ is quite difficult.

$$p\left(x\right) = \int p\left(x|z\right)p\left(z\right)dz$$

we can apply varitational inference to estimate this value.

Let's approximate $p(z|x)$ by another distribution $q(z|x)$ which we'll define such that it has a tractable distribution. If we can define the parameters of $q(z|x)$ such that it is very similar to $p(z|x)$, we can use it to perform approximate inference of the intractable distribution.

Recall that the KL divergence is a measure of difference between two probability distributions. Thus, if we wanted to ensure that $q(z|x)$ was similar to $p(z|x)$, we could minimize the KL divergence between the two distributions.
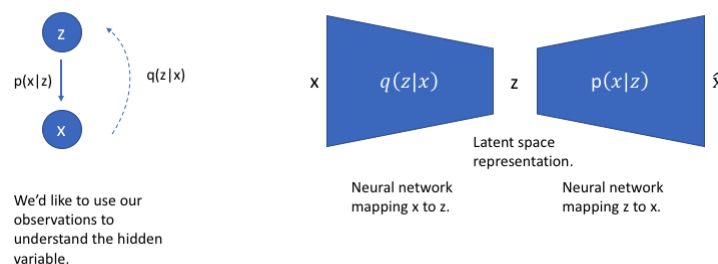
$$\min KL\left(q\left(z|x\right)||p\left(z|x\right)\right)$$

Dr. Ali Ghodsi goes through a full derivation here, but the result gives us that we can minimize the above expression by maximizing the following:

$$E_{q(z|x)}\log p\left(x|z\right) - KL\left(q\left(z|x\right)||p\left(z\right)\right)$$

The first term represents the reconstruction likelihood and the second term ensures that our learned distribution q is similar to the true prior distribution p.

To revisit our graphical model, we can use q to infer the possible hidden variables (ie. latent state) which was used to generate an observation. We can further construct this model into a neural network architecture where the encoder model learns a mapping from x to z and the decoder model learns a mapping from z back to x.

We'd like to use our observations to understand the hidden variable.

Latent space representation.

Neural network mapping x to z.

Neural network mapping z to x.

Our loss function for this network will consist of two terms, one which penalizes reconstruction error (which can be thought of maximizing the reconstruction likelihood as discussed earlier) and a second term which encourages our learned distribution $q(z|x)$ to be similar to the true prior distribution $p(z)$, which we'll assume follows a unit Gaussian distribution, for each

Implementation

In the previous section, I established the statistical motivation for a variational autoencoder structure. In this section, I'll provide the practical implementation details for building such a model yourself.

Rather than directly outputting values for the latent state as we would in a standard autoencoder, the encoder model of a VAE will output parameters describing a distribution for each dimension in the latent space. Since we're assuming that our prior follows a normal distribution, we'll output *two* vectors describing the mean and variance of the latent state distributions. If we were to build a true multivariate Gaussian model, we'd need to define a covariance matrix describing how each of the dimensions are correlated. However, we'll make a simplifying assumption that our covariance matrix only has nonzero values on the diagonal, allowing us to describe this information in a simple vector.

Our decoder model will then generate a latent vector by sampling from these defined distributions and proceed to develop a reconstruction of the original input.

However, this sampling process requires some extra attention. When training the model, we need to be able to calculate the relationship of each parameter in the network with respect to the final output loss using a technique known as backpropagation. However, we simply cannot do this for a *random sampling* process. Fortunately, we can leverage a clever idea known as the "reparameterization trick" which suggests that we randomly sample ε from a unit Gaussian, and then shift the randomly sampled ε by the latent distribution's mean μ and scale it by the latent distribution's variance σ.

With this reparameterization, we can now optimize the *parameters* of the distribution while still maintaining the ability to randomly sample from that distribution.

Note: In order to deal with the fact that the network may learn negative values for σ, we'll typically have the network learn

$\log \sigma$ *and exponentiate this value to get the latent distribution's variance.*

Visualization of latent space

To understand the implications of a variational autoencoder model and how it differs from standard autoencoder architectures, it's useful to examine the latent space. This blog post introduces a great discussion on the topic, which I'll summarize in this section.

The main benefit of a variational autoencoder is that we're capable of learning *smooth* latent state representations of the input data. For standard autoencoders, we simply need to learn an encoding which allows us to reproduce the input. As you can see in the left-most figure, focusing only on reconstruction loss *does* allow us to separate out the classes (in this case, MNIST digits) which should allow our decoder model the ability to reproduce the original handwritten digit, but there's an uneven distribution of data within the latent space. In other words, there are areas in latent space which don't represent *any* of our observed data.

Image credit (modified)

On the flip side, if we only focus only on ensuring that the latent distribution is similar to the prior distribution (through our KL divergence loss term), we end up describing *every* observation using the same unit Gaussian, which we subsequently sample from to describe the latent dimensions visualized. This effectively treats every observation as having the same characteristics; in other words, we've failed to describe the original data.

However, when the two terms are optimized simultaneously, we're encouraged to describe the latent state for an observation with distributions close to the prior but deviating when necessary to describe salient features of the input.

When I'm constructing a variational autoencoder, I like to inspect the latent dimensions for a few samples from the data to see the characteristics of the distribution. I encourage you to do the same.

If we observe that the latent distributions appear to be very tight, we may decide to give higher weight to the KL divergence term

broader distributions. This simple insight has led to the growth of a new class of models - disentangled variational autoencoders. As it turns out, by placing a larger emphasis on the KL divergence term we're also implicitly enforcing that the learned latent dimensions are uncorrelated (through our simplifying assumption of a diagonal covariance matrix).

$$\mathcal{L}\left(x, \hat{x}\right) + \beta \sum_{j} KL\left(q_j\left(z|x\right) || N\left(0, 1\right)\right)$$

Variational autoencoders as a generative model

By sampling from the latent space, we can use the decoder network to form a generative model capable of creating new data similar to what was observed during training. Specifically, we'll sample from the prior distribution $p\left(z\right)$ which we assumed follows a unit Gaussian distribution.

The figure below visualizes the data generated by the decoder network of a variational autoencoder trained on the MNIST handwritten digits dataset. Here, we've sampled a grid of values from a two-dimensional Gaussian and displayed the output of our decoder network.

As you can see, the distinct digits each exist in different regions of the latent space and smoothly transform from one digit to another. This smooth transformation can be quite useful when you'd like to interpolate between two observations, such as this recent example where Google built a model for interpolating between two music samples.
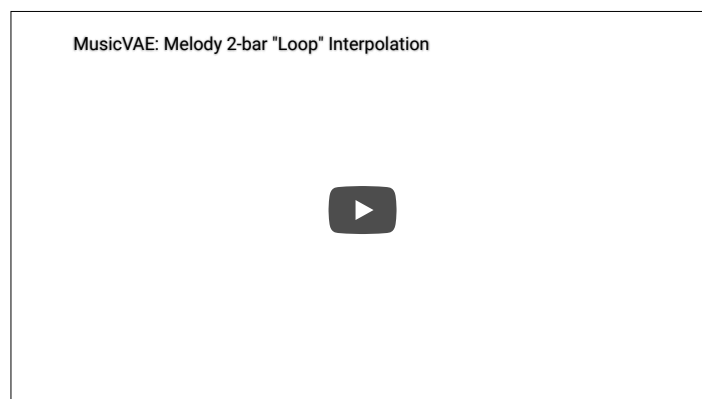
MusicVAE: Melody 2-bar "Loop" Interpolation

Further reading

- Ali Ghodsi: Deep Learning, Variational Autoencoder (Oct 12 2017)
- UC Berkley Deep Learning Decall Fall 2017 Day 6: Autoencoders and Representation Learning
- Stanford CS231n: Lecture on Variational Autoencoders

Blogs/videos

- Building Variational Auto-Encoders in TensorFlow (with great code examples)
- Building Autoencoders in Keras
- Variational Autoencoders - Arxiv Insights
- Intuitively Understanding Variational Autoencoders
- Density Estimation: A Neurotically In-Depth Look At Variational Autoencoders
- Pyro: Variational Autoencoders
- Under the Hood of the Variational Autoencoder
- With Great Power Comes Poor Latent Codes: Representation Learning in VAEs
- Kullback-Leibler Divergence Explained
- Neural Discrete Representation Learning

Papers/books

- Deep learning book (Chapter 20.10.3): Variational Autoencoders
- Variational Inference: A Review for Statisticians
- A tutorial on variational Bayesian inference
- Auto-Encoding Variational Bayes
- Tutorial on Variational Autoencoders

Papers on my reading list

- Neural Discrete Representation Learning
- Early Visual Concept Learning with Unsupervised Deep Learning
- Multimodal Unsupervised Image-to-Image Translation
    - Video from paper

You've successfully subscribed to Jeremy Jordan!

Subscribe to Jeremy Jordan

Jeremy Jordan — Variational autoencoders.   🐦   Subscribe

youremail@example.com   Subscribe

ALSO ON JEREMYJORDAN

Building machine learning products: a …	Gradient descent.	Introduction to autoencoders.	Evaluating a machine learning model.	Scaling near neighbors se
2 years ago • 1 comment	4 years ago	3 years ago • 6 comments	4 years ago • 2 comments	2 years ago • 2 co
Previously, I wrote about organizing machine learning projects where I …	Gradient descent is an optimization technique commonly used in …	Autoencoders are an unsupervised learning technique in which we …	So you've built a machine learning model and trained it on some data... now …	Jump to: What i neighbors searc Quantization Pro

28 Comments   JeremyJordan   🔒 Disqus' Privacy Policy   1 Login ▾

♡ Recommend 9   🐦 Tweet   f Share   Sort by Best ▾

👤   Join the discussion…

LOG IN WITH   OR SIGN UP WITH DISQUS ⍰

                         Name

👤   **Lester** • 3 years ago
Great post! Never understood why KL and reconstruction loss were used. The other thing I wonder about is why MSE is not as the reconstruction loss but rather I usually see binary cross-entropy applied to the pixels?
1 ∧ | ∨ • Reply • Share ›

   👤   **Jeremy Jordan** Mod ➜ Lester • 3 years ago
   That's a really good question! I believe it has to do with the nature (ie. distribution) of your data - Dr. Ghodsi alludes to this in his lecture.

   ▶

   2 ∧ | ∨ • Reply • Share ›

👤   **Asifa Junaid** • 19 days ago
wonderful post, article is written in simple way. I understand the concepts very well.Thanks a lot
∧ | ∨ • Reply • Share ›

👤   **Pietro Fusco** • a month ago • edited
This is an amazing post. I would kindly ask you if you can suggest or refer a post about Normalizing Flows. Thanks in advance.
∧ | ∨ • Reply • Share ›

👤   **parekh vivek** • a month ago
How to learn distribution numerical data? For examples, input parameters of housing price prediction dataset.
∧ | ∨ • Reply • Share ›

👤   **MM** • 4 months ago
Your blog was the first result on Google when I searched "Variational Autoencoder". Such wonderful explanations!
∧ | ∨ • Reply • Share ›

👤   **Omer** • 6 months ago
Hi, great post!
1 question if I may... You have written "When I'm constructing a variational autoencoder, I like to inspect the latent dimensions for a few samples from the data to see the characteristics of the distribution."
You mean taking a single specific sample x_i from the data and observing the histogram of the z_i's the encoder produces? I can simply look at the std or I'm mistaken?

You've successfully subscribed to Jeremy Jordan!   ✕

∧ | ∨ • Reply • Share ›

👤   **Wesley Neill** • 7 months ago

Jeremy Jordan – Variational autoencoders.   🐦   Subscribe

Weierstraß • a year ago

Hi, I really enjoy reading it! Thank you ! BTW, could you plz share the code for generating the images mentioned in the post?

⌃ | ⌄ • Reply • Share ›

Anandamoy Bandyopadhyay • 2 years ago

The best article on the net about VAEs that I've come across. Thanks a lot for this!

The KL term essentially tries to attract each of the latent feature means to zero and variance as close as to 1 as possible, right? However, for data with large variability does this restriction not adversely affect the geneartive performance of the model? Does it make sense to use $N(0, s)$ instead of $N(0, 1)$ where s is a data-dependent hyperparameter?

Also, can any differentiable image-image comparison measure be used in place of the reconstruction loss?

Would be much obliged if you could find the time to leave a reply!

⌃ | ⌄ • Reply • Share ›

Manjiro Kawakami • 2 years ago

Other 1 is here.

Please advise me about KL(q(z|x)||N(01)). Does P(z) represent classes of normal distributions with the coefficients with Maximum Likelihood? For example, the pictures of mountains, lakes, clouds, to be z1 and z2 and z3 being, say, 0.3, 0.5, and 0.2? Or, is this z simply the one to correspond to x, one by one in the reduced dimension space, with several Gaussian peaks to be confined in N(0,1)? I got lost. Please help.

⌃ | ⌄ • Reply • Share ›

Manjiro Kawakami • 2 years ago

2 questions more, thank you for many help. 1 is here.

Re-parametrization. With reconstruction loss, I thought it is basically, supervised training with backpropagation like CNN to Make X and X-bar identical. Why εnecessary? For each X, correspondent Z will be selected and Z will be distributed anyway. Where am I mistaken?

⌃ | ⌄ • Reply • Share ›

Manjiro Kawakami • 2 years ago

Thank you very much for your articles!

How you could put "SMILE" label in features for unsupervised learning?

I meaning is, you already labeling to training material with "SMILE" before VAE training, or labeling after VAE traning as put "SMILE" based on your judge of results freatures ?

⌃ | ⌄ • Reply • Share ›

> **Jeremy Jordan** Mod ➜ Manjiro Kawakami • 2 years ago
>
> Yeah good question, I put human-interpretable concepts for the dimensions in latent space so that people could understand what's going on. In reality, those dimensions are learned and can represent arbitrary concepts. A big interest in this area is learning a "disentangled" latent space where latent dimensions are uncorrelated and represent distinct concepts. When you train generative models, one thing you can do is search for a vector in the latent space which does represent a concept, and modify the values along this vector to adjust an explicit concept.
>
> ⌃ | ⌄ • Reply • Share ›
>
> > **Manjiro Kawakami** ➜ Jeremy Jordan • 2 years ago
> >
> > Thank you again. Questions. Can I understand that we can input X into Encoder randomly without categorizing smile, glass, color-of-skin, etc, in advance, right?
> >
> > I naturally presume such features like "smile" cannot be coincidentally eigenvalue (in case of PCA). Do you mean that we can tame the feature to be independent (diagonal) by tampering somehow?
> >
> > ⌃ | ⌄ • Reply • Share ›
> >
> > **Manjiro Kawakami** ➜ Jeremy Jordan • 2 years ago
> >
> > Thank you very much for your prompt reply, exciting.
> >
> > I have a interesting what is the "human-interpretable concepts" in actual program but I trying to understand your advice detail first. Again, thank you very much for your articles and reply !
> >
> > ⌃ | ⌄ • Reply • Share ›

issy • 2 years ago

hi jeremy, if we take a random encoding and do nearest neighbor search, do we end up with similar neighbors? i don't see many similars though.

⌃ | ⌄ • Reply • Share ›

> **Jeremy Jordan** Mod ➜ issy • 2 years ago
>
> what do you mean by random encoding? a randomly selected image (and its encoding) from your dataset or a randomly generated vector? can you provide more details surrounding the problem you're working on?
>
> ⌃ | ⌄ • Reply • Share ›
>
> > **issy** ➜ Jeremy Jordan • 2 years ago • edited
> >
> > my goal is to get similar images clustered. So, when i do a nearest neighbor search, i could find all the visually similars as nearest neighbors. I train a VAE , throw away the decoder. Then select a random image from dataset and its encoding and look for nearest neighbors. I see that there are lot of neighbors which belong to different classes as the clusters in latent representation are close to each other. I was thinking that could be because of KL divergence as it tries to maintain that overlap between clusters so that the decoder can easily interpolate.
> > Training the model and throwing away the decoder and using the encoder seems like just doing the PCA and knn.
> >
> > ⌃ | ⌄ • Reply • Share ›
> >
> > > **Jeremy Jordan** Mod ➜ issy • 2 years ago • edited
> > >
> > > if you have a good VAE model then the encoding space should capture a visual similarity metric. for example, notice how model trained on MNIST data has a smooth latent space with similar numbers (eg. 1 and 7) occurring nearby. your hypothesis makes sense, have you tried adjusting the hyperparameter to balance between reconstruction loss and KL divergence?
> > >
> > > it also sounds like this paper https://arxiv.org/abs/1907.... would be a good approach to explore for your problem.
> > >
> > > ⌃ | ⌄ • Reply • Share ›
> > >
> > > **issy** ➜ Jeremy Jordan • 2 years ago

Mithi Sevilla • 2 years ago • edited

Hi Jeremy! Thanks for writing this well-explained article about VAEs . I included it in this little project of mine called 📷📄 Deep

🐦 Subscribe

add it!
⌃ | ⌄ • Reply • Share ›

Mostafa Hussien • 2 years ago
Actually a great post and very clear explanation for the steps of VAE in addition to the great simplification for the concepts. I need to told you that this is the 10th tutorial i read about VAE and this is the only one that makes me understand the concept well.
Thanks
⌃ | ⌄ • Reply • Share ›

dan pang (pangdan) • 2 years ago
Thanks for great post! I am wonder why we use q(z) to approximate p(z|x) rather than q(z|x) in Dr Ghodsi's video.
⌃ | ⌄ • Reply • Share ›

Jeremy Jordan Mod ➜ dan pang (pangdan) • 2 years ago
In Dr. Ghodsi's video he mentions at 23:34 that it depends on your interpretation of the problem.
⌃ | ⌄ • Reply • Share ›

Sia Rez • 2 years ago
For anyone who is interested in playing around with an in-browser interactive demo of the VAE, check out my post:
https://www.siarez.com/proj...
⌃ | ⌄ • Reply • Share ›

MORE IN **DATA SCIENCE**

A simple solution for monitoring ML systems.
2 Jan 2021 – 10 min read

Effective testing for machine learning systems.
19 Aug 2020 – 9 min read

An introduction to Kubernetes.
26 Nov 2019 – 15 min read

See all 47 posts →

DATA SCIENCE

Common architectures in convolutional neural networks.

In this post, I'll discuss commonly used architectures for convolutional networks. As you'll see, almost all CNN architectures follow the same general design principles of successively applying convolutional layers to the input, periodically downsampling the spatial dimensions while increasing the number of feature maps.

JEREMY JORDAN
19 APR 2018 • 9 MIN READ

DATA SCIENCE

Introduction to autoencoders.

Autoencoders are an unsupervised learning technique in which we leverage neural networks for the task of representation learning. Specifically, we'll design a neural network architecture such that we impose a bottleneck in the network which forces a compressed knowledge representation of the original input.

JEREMY JORDAN
19 MAR 2018 • 10 MIN READ

Jeremy Jordan © 2021

Latest Posts · Twitter · Ghost