



# Understanding Vector Quantized Variational Autoencoders (VQ-VAE)



Shashank Yadav · [Follow](#)

5 min read · Sep 1, 2019



482

10



From my most recent escapade into the deep learning literature I present to you this paper by [Oord et. al.](#) which presents the idea of **using discrete latent embeddings for variational auto encoders**. The proposed model is called Vector Quantized Variational Autoencoders (VQ-VAE). I really liked the idea and the results that came with it but found surprisingly few resources to develop an understanding. Here's an attempt to help other who might venture into this domain after me.

Like numerous other people Variational Autoencoders (VAEs) are my choice of generative models. Unlike GANs they are easier to train and reason about (No offence intended dear GANs). Going forward I assume you have some understanding of VAEs. If you don't I suggest going through [this post](#), I found it to be one of the simpler ones.

## Basic Idea

So what is the big deal here? As you might recall, VAEs consist of 3 parts:

1. An encoder network that parametrizes the posterior  $q(z|x)$  over latents
2. A prior distribution  $p(z)$
3. A decoder with distribution  $p(x|z)$  over input data

Typically we assume this prior and posterior to be normally distributed with diagonal variance. The encoder is then used to predict the mean and variances of the posterior.

In the proposed work however, the authors use discrete latent variables (instead of a continuous normal distribution). The posterior and prior distributions are categorical, and the samples drawn from these distributions index an embedding table. In other words:

1. Encoders model a categorical distribution, sampling from which you get integral values
2. These integral values are used to index a dictionary of embeddings
3. The indexed values are then passed on to the decoder

## Why do it?

Many important real-world objects are discrete. For example in images we might have categories like “Cat”, “Car”, etc. and it might not make sense to interpolate between these categories. Discrete representations are also easier to model since each category has a single value whereas if we had a continuous latent space then we will need to normalize this density function and learn the dependencies between the different variables which could be very complex.

Moreover, the authors claim that their model doesn't suffer from posterior collapse, an issue that plagues VAEs in general and prevents making use of complex decoders.

## Architecture

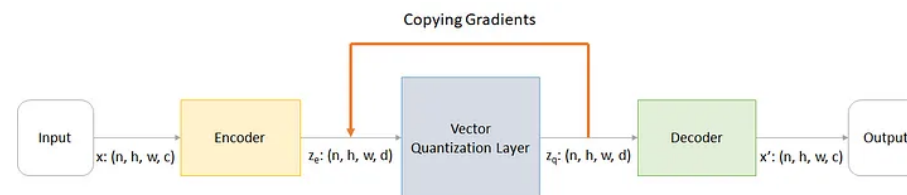


Fig 1: VQ-VAE Architecture

Fig 1 shows various top level components in the architecture along with dimensions at each step. Assuming we run our model over image data, here's some nomenclature we'll be using going forward:

n : batch size

h: image height

w: image width

c: number of channels in the input image

d: number of channels in the hidden state

Now the working can be explained in the following steps:

1. Encoder takes in images  $x$ :  $(n, h, w, c)$  and give outputs  $z_e$ :  $(n, h, w, d)$
2. Vector Quantization layer takes  $z_e$  and selects embeddings from a dictionary based on distance and outputs  $z_q$  (we'll discuss more about this later don't worry)
3. Decoder consumes  $z_q$  and outputs  $x'$  trying to recreate input  $x$

Top highlight

## Vector Quantization Layer

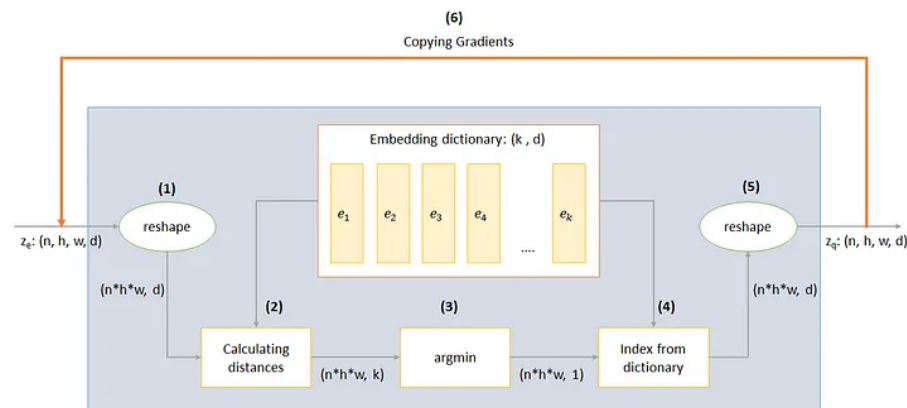


Fig 2: Vector Quantization Layer

The working of VQ layer can be explained in six steps as numbered in Fig 2:

1. **Reshape:** all dimensions except the last one are combined into one so that we have  $n \times h \times w$  vectors each of dimensionality  $d$
2. **Calculating distances:** for each of the  $n \times h \times w$  vectors we calculate distance from each of  $k$  vectors of the embedding dictionary to obtain a matrix of shape  $(n \times h \times w, k)$
3. **Argmin:** for each of the  $n \times h \times w$  vectors we find the index of closest of the  $k$  vectors from dictionary
4. **Index from dictionary:** index the closest vector from the dictionary for each of  $n \times h \times w$  vectors
5. **Reshape:** convert back to shape  $(n, h, w, d)$
6. **Copying gradients:** If you followed up till now you'd realize that it's not possible to train this architecture through backpropagation as the gradient won't flow through argmin. Hence we try to approximate by copying the gradients from  $z_q$  back to  $z_e$ . In this way we're not actually minimizing the loss function but are still able to pass some information back for training.

### Loss Function

The total loss is actually composed of three components:

1. **Reconstruction loss:** which optimizes the decoder and encoder:

$$\text{reconstruction\_loss} = -\log( p(x|z_q) )$$

2. **Codebook loss:** due to the fact that gradients bypass the embedding, we use a dictionary learning algorithm which uses an  $l_2$  error to move the embedding vectors  $e_i$  towards the encoder output:

```
codebook_loss = || sg[z_e(x)] - e ||^2
// sg represents stop gradient operator meaning no gradient
// flows through whatever it's applied on
```

**3. Commitment loss:** since the volume of the embedding space is dimensionless, it can grow arbitrarily if the embeddings  $e_i$  do not train as fast as the encoder parameters, and thus we add a commitment loss to make sure that the encoder commits to an embedding

```
commitment_loss =  $\beta$  ||  $z_e(x) - \text{sg}[e]$  ||2  
//  $\beta$  is a hyperparameter that controls how much we want to weigh  
// commitment loss compared to other components
```

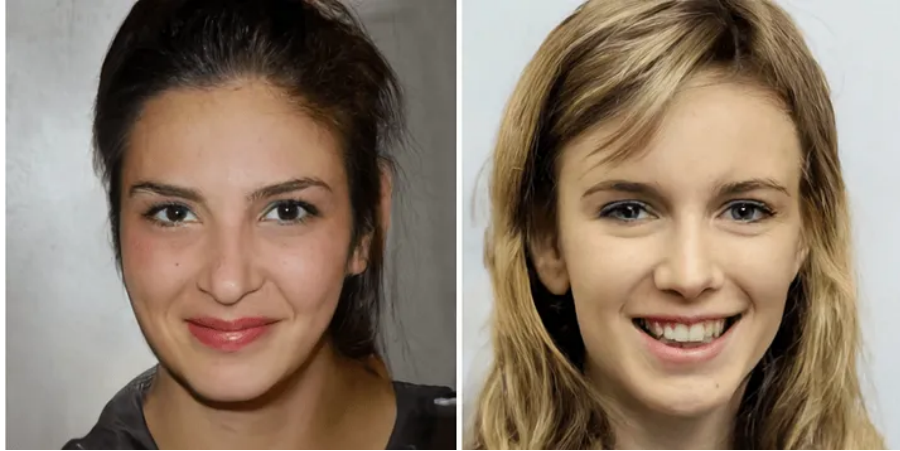
***Important:** Note that we're training both the dictionary embeddings as well as encoder and decoder network*

## Results

The paper presents state of the art results on images, text as well as videos.



VQ-VAE samples (left) and BigGAN deep samples (right) trained on ImageNet.



VQ-VAE generated facial images.

You can find the results on audio here:

<https://avdnoord.github.io/homepage/vqvae/>

## Code

There are several implementations available in Tensorflow, Pytorch as well as keras. You can look through them [here](#).

## Conclusion

There are two main ideas to be learnt from this paper:

1. How to train discrete latent embeddings and their importance
2. How to approximate gradients in case of non differentiable functions

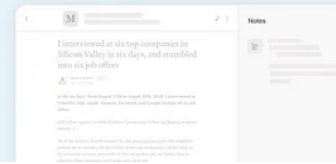
For more details go through the paper, it'd be easier to understand after going through the article. You can also play around with [this Jupyter Notebook](#), open it in colab [here](#). Happy learning!

---


***Note:** Feel free to ask any doubts or give feedback/suggestions. All the diagrams used here have been created by the author. Feel free to use them along with a note of acknowledgement :-)*

Read this story later in Journal.

Meet Journal →



 Read this story later in Journal.

 Wake up every Sunday morning to the week's most noteworthy stories in Tech waiting in your inbox. Read the Noteworthy in Tech newsletter.

Machine Learning

Deep Learning

Variational Autoencoder

Generative Adversarial

Generative Model



**Written by Shashank Yadav**

111 Followers

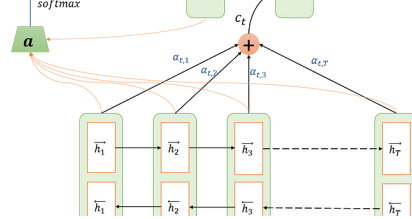
Founding [fractionai.xyz](#): decentralized data labelling platform


Follow



---

More from Shashank Yadav



 Shashank Yadav

## Understanding Attention Mechanism

Attention mechanism for sequence modelling was first introduced in the paper: Neural...

5 min read · Feb 6, 2019



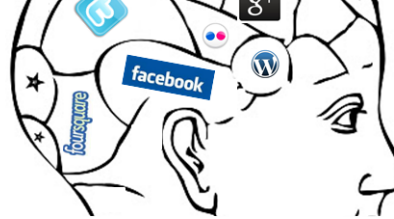
454



5



...



 Victoria Halina

## The Psychology of Social Media— Why We Feel the Need to Share

The phenomenon that altered the relationship we have with others and...

★ · 4 min read · Jan 31, 2019



2.3K



17



...



 Holly

## The Problem with Referring to Women as “Females”

Although the word holds scientific truth, social construction had rendered it a...

★ · 4 min read · Feb 20, 2020




412



18



...

 Shashank Yadav in Analytics Vidhya

## Introducing Fastpipeline for ML

TLDR; Quick and easy pipelines that autodetect duplicate runs and reuse...

5 min read · Dec 5, 2020



7

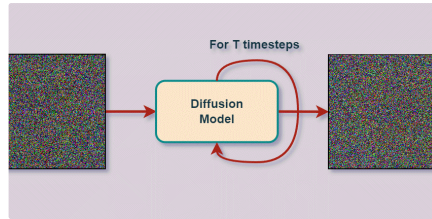



...

See all from Shashank Yadav



## Recommended from Medium



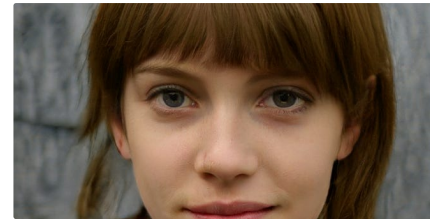
 Gabriel Mongaras in Better Programming

### Diffusion Models—DDPMs, DDIMs, and Classifier Free Guidance

A guide to the evolution of diffusion models from DDPMs to Classifier Free guidance

28 min read · Mar 13, 2023

 616  8



 Everton Gomedé, PhD

### StyleGAN: Revolutionizing the Art of Generative Adversarial...

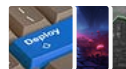
Introduction

5 min read · Nov 10, 2023

 5 



## Lists



### Predictive Modeling w/ Python

20 stories · 819 saves



### Practical Guides to Machine Learning

10 stories · 949 saves



### Natural Language Processing

1117 stories · 588 saves




### data science and AI

39 stories · 49 saves



 Aguiamar Neto



 Will Badr  in Towards Data Science

## What is Latent Diffusion in AI?

Latent diffusion models are deep learning models that have recently emerged as a...

5 min read · Oct 7, 2023



Harshita Sharma in Accredian

## Building Intuition: Variational Autoencoders (VAEs)

How Variational Autoencoders are able to generate new data

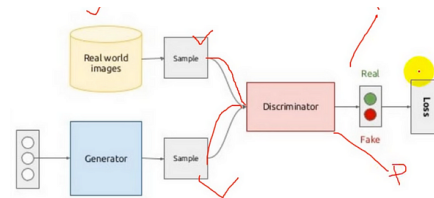
7 min read · Sep 25, 2023



## Uncovering Anomalies with Variational Autoencoders (VAE): ...

An example use case of using Variational Autoencoders (VAE) to detect anomalies in ...

🌟 · 9 min read · Jan 17, 2023



Computer Science Engineering

## Generative Adversarial Network GAN

A generative adversarial network is a class of machine learning frameworks invented by la...

14 min read · Jul 25, 2023



See more recommendations