A General And Adaptive Loss Function

Basics of numerical analysis

- Sina Tayebi
- Mahdi Ahmadi
- Ariana Aghamohamadi

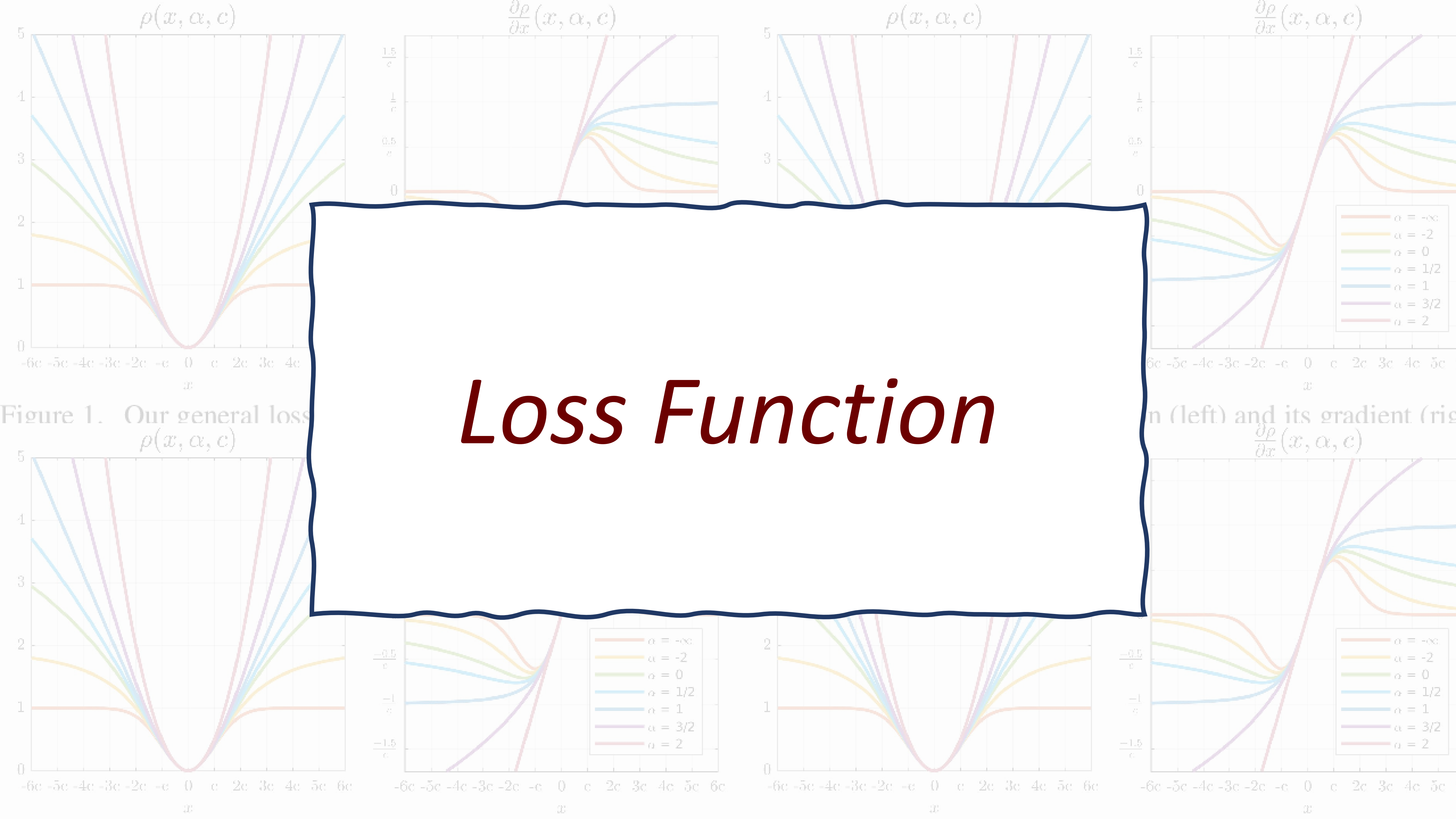
Introduction:

We present a single loss function that is a superset of many common robust loss functions.
A single continuous-valued parameter in our general loss function can be set such that it is equal to several traditional losses, and can be adjusted to model a wider family of functions.
This allows us to generalize algorithms built around a fixed robust loss with a new "robustness" hyperparameter that can be tuned or annealed to improve performance.

Content:

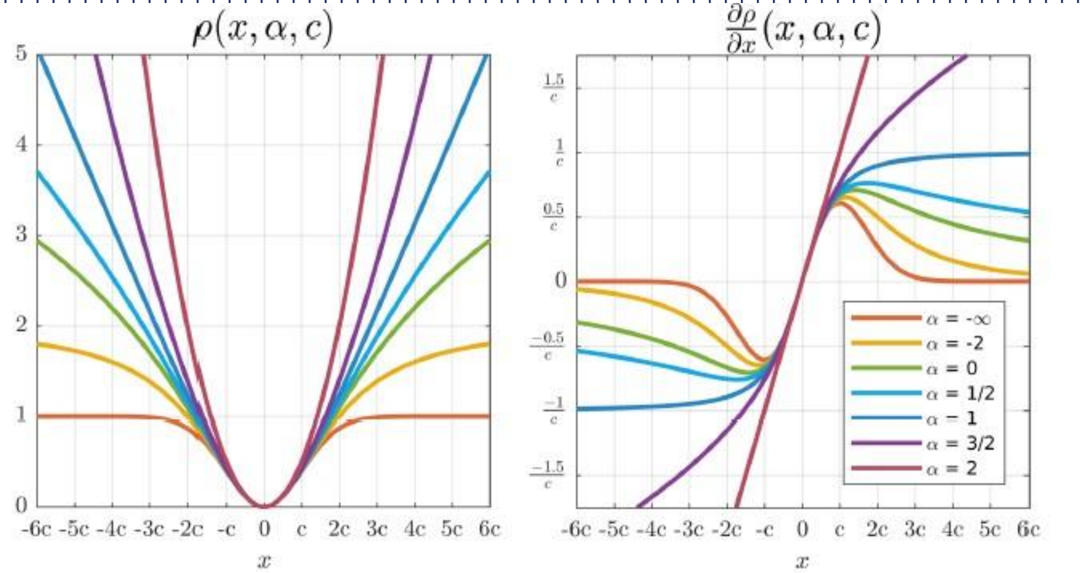
- Introducing loss function
- Density probability
- Experiments
- Code implement

Loss Function



The general form of loss function:

$$f\left(x,\alpha,c\right)=\frac{|\alpha-2|}{\alpha}\left(\left(\frac{\left({}^{x}/_{c}\right)^{2}}{|\alpha-2|}+1\right)^{\alpha/2}-1\right)$$



Derivatives:

$$\frac{\partial\rho}{\partial x}\left(x,\alpha,c\right)=\begin{cases}\frac{x}{c^{2}} & \text{if } \alpha=2\\ \frac{2x}{x^{2}+2c^{2}} & \text{if } \alpha=0\\ \frac{x}{c^{2}}\exp\left(-\frac{1}{2}\left({}^{x}/_{c}\right)^{2}\right) & \text{if } \alpha=-\infty\\ \frac{x}{c^{2}}\left(\frac{\left({}^{x}/_{c}\right)^{2}}{|\alpha-2|}+1\right)^{\left(\alpha/2-1\right)} & \text{otherwise}\end{cases}$$

Different alpha values:

$$\rho\left(x,\alpha,c\right)=\begin{cases}\frac{1}{2}\left({}^{x}/_{c}\right)^{2} & \text{if } \alpha=2\\ \log\left(\frac{1}{2}\left({}^{x}/_{c}\right)^{2}+1\right) & \text{if } \alpha=0\\ 1-\exp\left(-\frac{1}{2}\left({}^{x}/_{c}\right)^{2}\right) & \text{if } \alpha=-\infty\\ \frac{|\alpha-2|}{\alpha}\left(\left(\frac{\left({}^{x}/_{c}\right)^{2}}{|\alpha-2|}+1\right)^{\alpha/2}-1\right) & \text{otherwise}\end{cases}$$

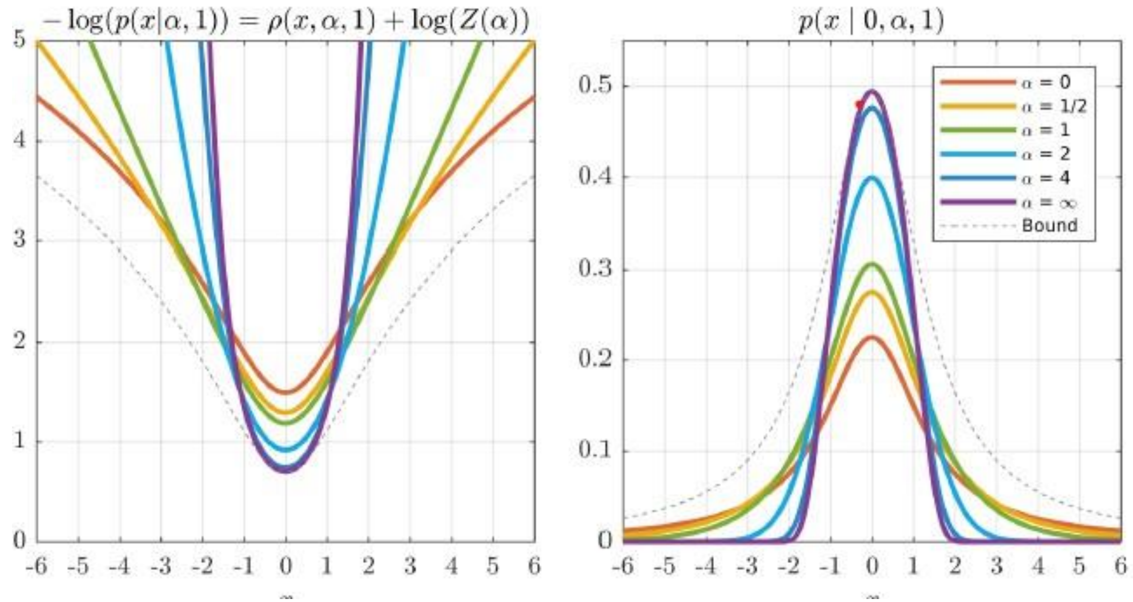
Probability Density Function(PDF)

Just as our loss function includes several common loss functions as special cases, our distribution includes several common distributions as special cases.

$$p(x \mid \mu, \alpha, c) = \frac{1}{cZ(\alpha)} \exp\left(-\rho(x - \mu, \alpha, c)\right)$$

α = 2 => normal guassian

α = 0 => Cauchy

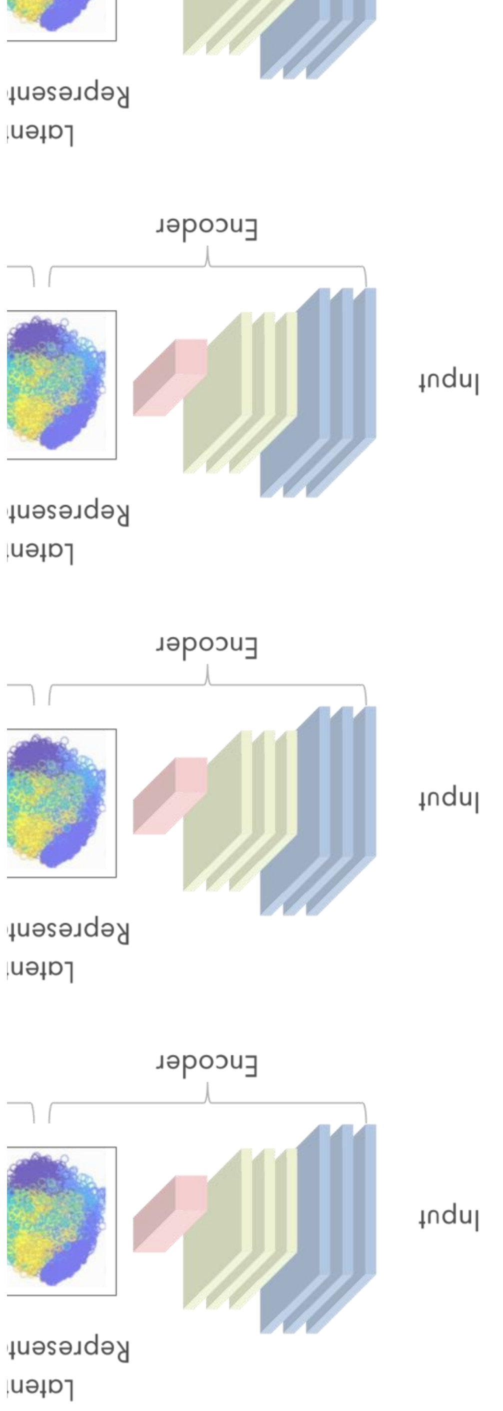


Experiments

In this section we try to demonstrate the utility of our loss and distribution with 4 experiments

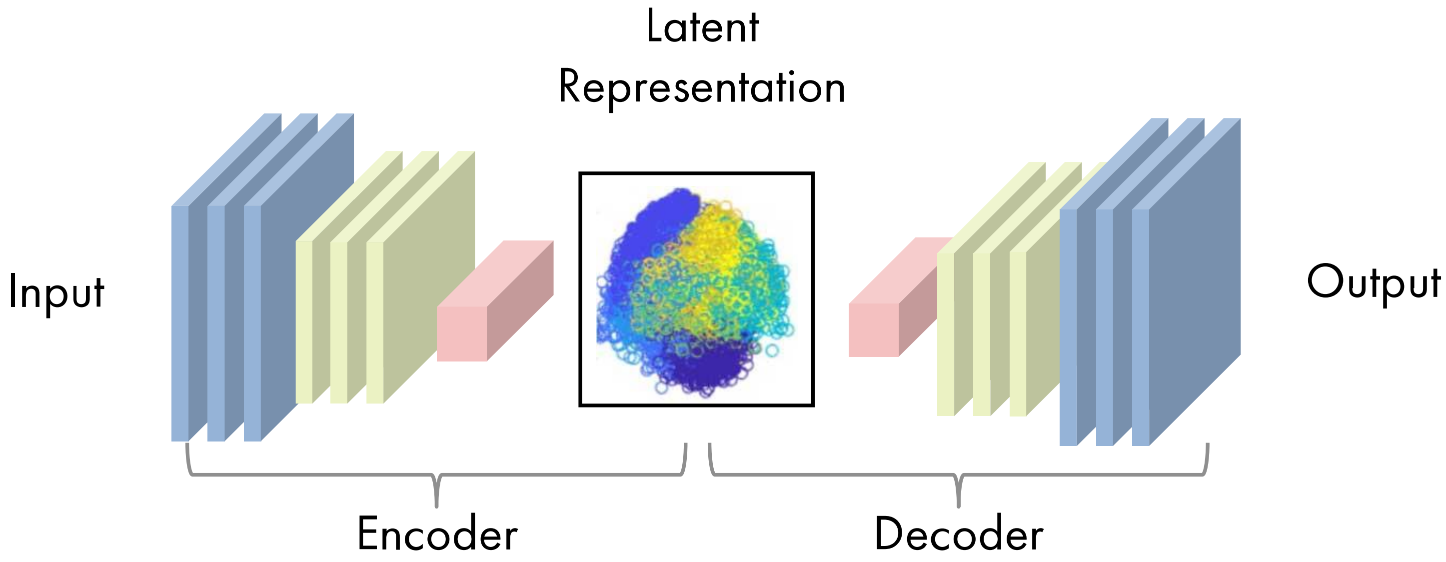
We will show that across a variety of tasks, just replacing the loss function of an existing model with our general loss function can enable significant performance improvements.

1-Variational Autoencoders:



What Are Autoencoders?

Autoencoders are neural networks containing an encoder and a decoder that learn to compress input data into a lower-dimensional representation with the encoder and then reconstruct it back to the original data space with its decoder .

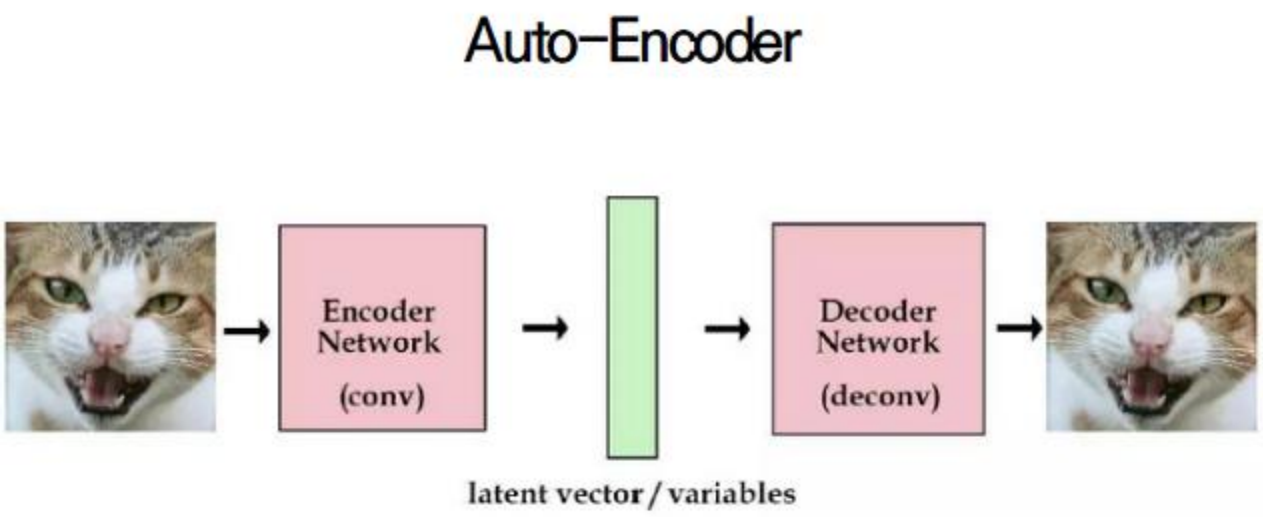


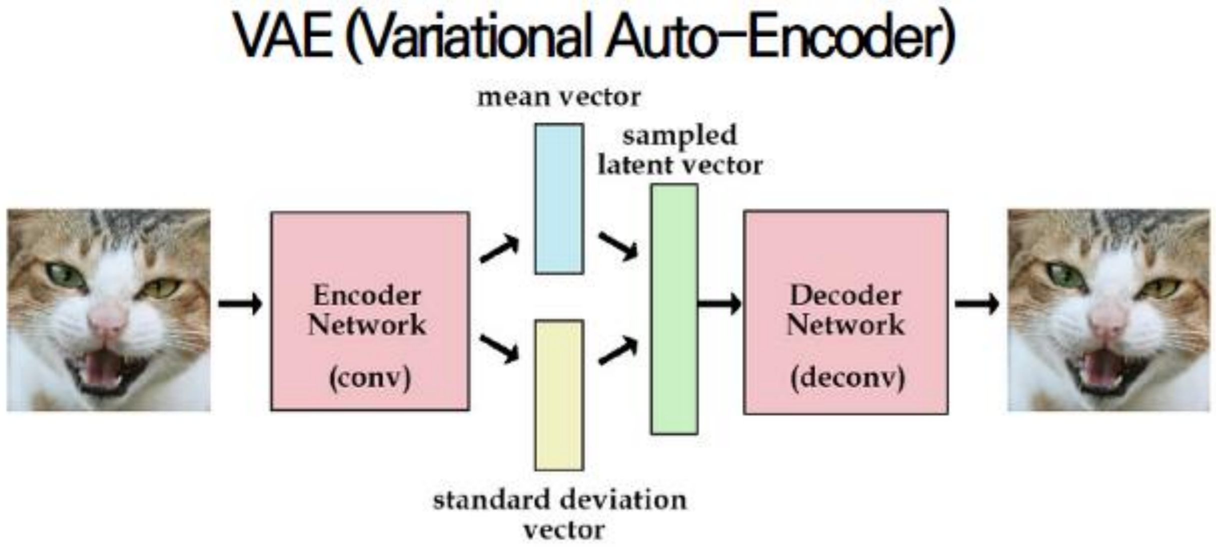
What Are *Variational* Autoencoders?

Autoencoders that can generate new images are called variational autoencoders.

instead of trying to perfectly reconstruct the input image, the VAE tries to learn a probability distribution over the possible latent codes that could have generated the input image.

A common design decision for VAEs is to model images using an independent normal distribution on a vector of RGB pixel values, and we use this design as our baseline model.





Why does this exist?

- Learn a hidden representation of input (that "vector")
- Can NOT generate new data

What does this optimize?

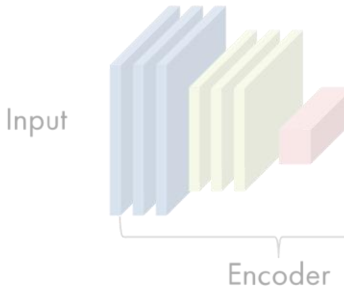
- Minimize reconstruction loss

- Learns to generate new data
- Minimize reconstruction loss + latent loss
- Latent vectors are sampled from Gaussian Mixture

we will explore the hypothesis that the baseline model of normal distributions placed on a per-pixel image representation can be improved significantly with the small change of just modeling a linear transformation of a VAE's output with our general distribution.

We compare the results of 4 distributions:

Normal - Cauchy - t-dist - our general dist.



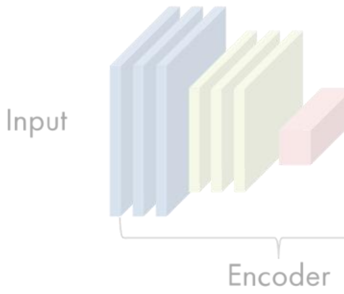
Training with our general distribution:

$$\alpha^{(i)} = (\alpha_{\max} - \alpha_{\min})\,\mathrm{sigmoid}\left(\alpha_\ell^{(i)}\right) + \alpha_{\min}$$

$$c^{(i)} = \mathrm{softplus}\left(c_\ell^{(i)}\right) + c_{\min}$$

$$\alpha_{\min} = 0,\ \ \alpha_{\max} = 3,\ \ c_{\min} = 10^{-8}$$

Alpha is a shape parameter and c is a scale parameter remaining from the normal distribution



Result:

For this, we used the DCT and the CDF 9/7 wavelet decomposition, both with a YUV color space.

We can observe that our general loss function did a better job than the other 3 distributions in 2 out of 3 image representations

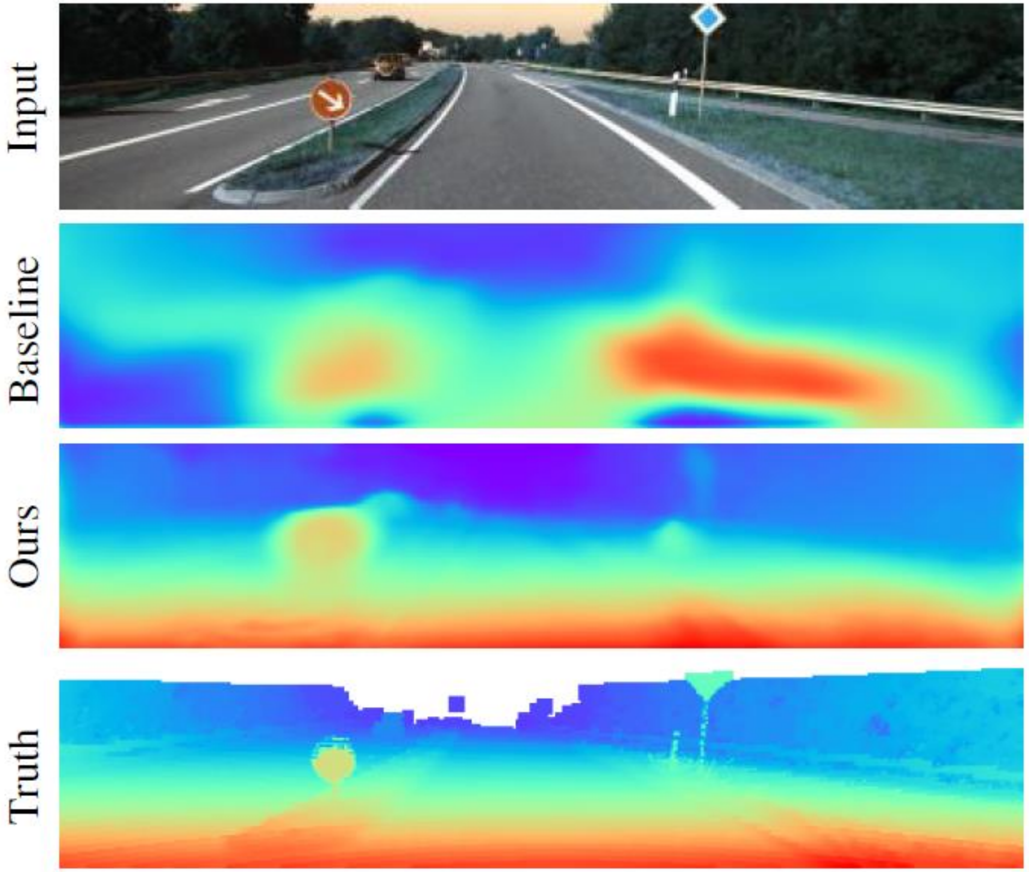
	Normal	Cauchy	t-dist.	Ours
Pixels + RGB	8,662	9,602	10,177	10,240
DCT + YUV	31,837	31,295	32,804	32,806
Wavelets + YUV	31,505	35,779	36,373	36,316

2_ Unsupervised Monocular Depth Estimation:

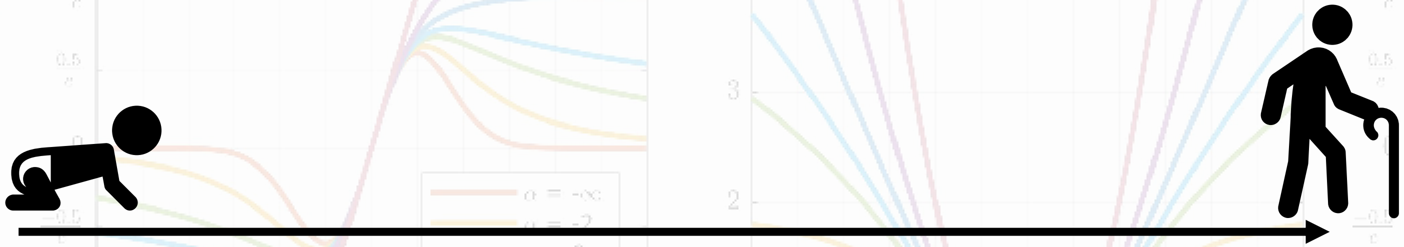
- This model is trained by minimizing the differences between two images in a stereo pair.
- the difference between images is defined as the absolute difference between RGB values.
- The absolute loss is equivalent to maximizing the likelihood of a Laplacian distribution with a fixed scale on RGB pixel values.
- We replace that fixed Laplacian distribution with our general distribution, keeping our scale fixed but allowing the shape parameter to vary.
- All training and evaluation were performed on the KITTI dataset.

Result:

We can observe adaptive alpha outperforms fixed and annealing alphas with different metrics and they all perform better than the original loss function for the model.



		lower is better				higher is better		
	Avg	AbsRel	SqRel	RMS	logRMS	<1.25	$<1.25^2$	$<1.25^3$
Baseline [42] as reported	0.407	0.221	2.226	7.527	0.294	0.676	0.885	0.954
Baseline [42] reproduced	0.398	0.208	2.773	7.085	0.286	0.726	0.895	0.953
Ours, fixed $\alpha = 1$	0.356	0.194	2.138	6.743	0.268	0.738	0.906	0.960
Ours, fixed $\alpha = 0$	0.350	0.187	2.407	6.649	0.261	0.766	0.911	0.960
Ours, fixed $\alpha = 2$	0.349	0.190	1.922	6.648	0.267	0.737	0.904	0.961
Ours, annealing $\alpha = 2 \to 0$	0.341	0.184	2.063	6.697	0.260	0.756	0.911	0.963
Ours, adaptive $\alpha \in (0, 2)$	0.332	0.181	2.144	6.454	0.254	0.766	0.916	0.965



The End