A General and Adaptive Robust Loss Function

Jonathan T. Barron
Google Research

Abstract

We present a generalization of the Cauchy/Lorentzian, Geman-McClure, Welsch/Leclerc, generalized Charbonnier, Charbonnier/pseudo-Huber/L1-L2, and L2 loss functions. By introducing robustness as a continuous parameter, our loss function allows algorithms built around robust loss minimization to be generalized, which improves performance on basic vision tasks such as registration and clustering. Interpreting our loss as the negative log of a univariate density yields a general probability distribution that includes normal and Cauchy distributions as special cases. This probabilistic interpretation enables the training of neural networks in which the robustness of the loss automatically adapts itself during training, which improves performance on learning-based tasks such as generative image synthesis and unsupervised monocular depth estimation, without requiring any manual parameter tuning.

Many problems in statistics and optimization require *robustness* — that a model be less influenced by outliers than by inliers [17, 19]. This idea is common in parameter estimation and learning tasks, where a robust loss (say, absolute error) may be preferred over a non-robust loss (say, squared error) due to its reduced sensitivity to large errors. Researchers have developed various robust penalties with particular properties, many of which are summarized well in [3, 39]. In gradient descent or M-estimation [16] these losses are often interchangeable, so researchers may experiment with different losses when designing a system. This flexibility in shaping a loss function may be useful because of non-Gaussian noise, or simply because the loss that is minimized during learning or parameter estimation is different from how the resulting learned model or estimated parameters will be evaluated. For example, one might train a neural network by minimizing the difference between the network's output and a set of images, but evaluate that network in terms of how well it hallucinates random images.

In this paper 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.

Though new hyperparameters may be valuable to a practitioner, they complicate experimentation by requiring manual tuning or time-consuming cross-validation. However, by viewing our general loss function as the negative log-likelihood of a probability distribution, and by treating the robustness of that distribution as a latent variable, we show that maximizing the likelihood of that distribution allows gradient-based optimization frameworks to *automatically* determine how robust the loss should be without any manual parameter tuning. This "adaptive" form of our loss is particularly effective in models with multivariate output spaces (say, image generation or depth estimation) as we can introduce independent robustness variables for each dimension in the output and thereby allow the model to independently adapt the robustness of its loss in each dimension.

The rest of the paper is as follows: In Section 1 we define our general loss function, relate it to existing losses, and enumerate some of its useful properties. In Section 2 we use our loss to construct a probability distribution, which requires deriving a partition function and a sampling procedure. Section 3 discusses four representative experiments: In Sections 3.1 and 3.2 we take two

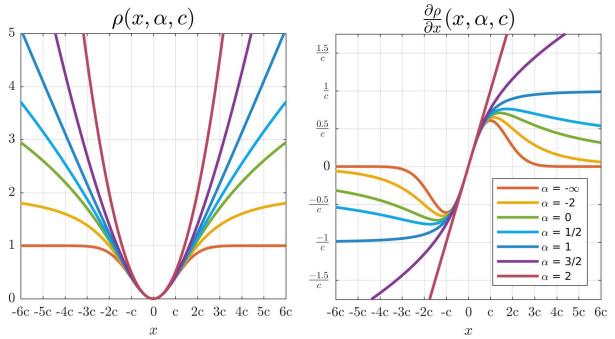

Figure 1. Our general loss function (left) and its gradient (right) for different values of its shape parameter α. Several values of α reproduce existing loss functions: L2 loss ($\alpha = 2$), Charbonnier loss ($\alpha = 1$), Cauchy loss ($\alpha = 0$), Geman-McClure loss ($\alpha = -2$), and Welsch loss ($\alpha = -\infty$).

vision-oriented deep learning models (variational autoencoders for image synthesis and self-supervised monocular depth estimation), replace their losses with the negative log-likelihood of our general distribution, and demonstrate that allowing our distribution to automatically determine its own robustness can improve performance without introducing any additional manually-tuned hyperparameters. In Sections 3.3 and 3.4 we use our loss function to generalize algorithms for the classic vision tasks of registration and clustering, and demonstrate the performance improvement that can be achieved by introducing robustness as a hyperparameter that is annealed or manually tuned.

1. Loss Function

The simplest form of our loss function is:

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

Here $\alpha\in\mathbb{R}$ is a shape parameter that controls the robustness of the loss and $c>0$ is a scale parameter that controls the size of the loss's quadratic bowl near $x=0$.

Though our loss is undefined when $\alpha=2$, it approaches L2 loss (squared error) in the limit:

$$\lim_{\alpha\to 2} f\left(x,\alpha,c\right)=\frac{1}{2}\left(x/c\right)^2 \tag{2}$$

When $\alpha=1$ our loss is a smoothed form of L1 loss:

$$f\left(x,1,c\right)=\sqrt{\left(x/c\right)^2+1}-1 \tag{3}$$

This is often referred to as Charbonnier loss [5], pseudo-Huber loss (as it resembles Huber loss [18]), or L1-L2 loss [39] (as it behaves like L2 loss near the origin and like L1 loss elsewhere).

Our loss's ability to express L2 and smoothed L1 losses is shared by the "generalized Charbonnier" loss [34], which has been used in flow and depth estimation tasks that require robustness [6, 23] and is commonly defined as:

$$\left(x^2+\epsilon^2\right)^{\alpha/2} \tag{4}$$

Our loss has significantly more expressive power than the generalized Charbonnier loss, which we can see by setting our shape parameter α to nonpositive values. Though $f\left(x,0,c\right)$ is undefined, we can take the limit of $f\left(x,\alpha,c\right)$ as α approaches zero:

$$\lim_{\alpha\to 0} f\left(x,\alpha,c\right)=\log\left(\frac{1}{2}\left(x/c\right)^2+1\right) \tag{5}$$

This yields Cauchy (aka Lorentzian) loss [2]. By setting $\alpha=-2$, our loss reproduces Geman-McClure loss [13]:

$$f\left(x,-2,c\right)=\frac{2\left(x/c\right)^2}{\left(x/c\right)^2+4} \tag{6}$$

In the limit as α approaches negative infinity, our loss becomes Welsch [20] (aka Leclerc [25]) loss:

$$\lim_{\alpha\to-\infty} f\left(x,\alpha,c\right)=1-\exp\left(-\frac{1}{2}\left(x/c\right)^2\right) \tag{7}$$

With this analysis we can present our final loss function, which is simply $f\left(\cdot\right)$ with special cases for its removable singularities at $\alpha=0$ and $\alpha=2$ and its limit at $\alpha=-\infty$.

$$\rho\left(x,\alpha,c\right)=\begin{cases}\frac{1}{2}\left(x/c\right)^2 & \text{if } \alpha=2\\ \log\left(\frac{1}{2}\left(x/c\right)^2+1\right) & \text{if } \alpha=0\\ 1-\exp\left(-\frac{1}{2}\left(x/c\right)^2\right) & \text{if } \alpha=-\infty\\ \frac{\left|\alpha-2\right|}{\alpha}\left(\left(\frac{\left(x/c\right)^2}{\left|\alpha-2\right|}+1\right)^{\alpha/2}-1\right) & \text{otherwise}\end{cases} \tag{8}$$

As we have shown, this loss function is a superset of the Welsch/Leclerc, Geman-McClure, Cauchy/Lorentzian, generalized Charbonnier, Charbonnier/pseudo-Huber/L1-L2, and L2 loss functions.

To enable gradient-based optimization we can derive the derivative of $\rho\left(x,\alpha,c\right)$ with respect to x:

$$\frac{\partial\rho}{\partial x}\left(x,\alpha,c\right)=\begin{cases}\frac{x}{c^2} & \text{if } \alpha=2\\ \frac{2x}{x^2+2c^2} & \text{if } \alpha=0\\ \frac{x}{c^2}\exp\left(-\frac{1}{2}\left(x/c\right)^2\right) & \text{if } \alpha=-\infty\\ \frac{x}{c^2}\left(\frac{\left(x/c\right)^2}{\left|\alpha-2\right|}+1\right)^{\left(\alpha/2-1\right)} & \text{otherwise}\end{cases} \tag{9}$$

Our loss and its derivative are visualized for different values of α in Figure 1.

The shape of the derivative gives some intuition as to how α affects behavior when our loss is being minimized by gradient descent or some related method. For all values of α the derivative is approximately linear when $|x|<c$, so the effect of a small residual is always linearly proportional to that residual's magnitude. If $\alpha=2$, the derivative's magnitude stays linearly proportional to the residual's magnitude — a larger residual has a correspondingly larger effect. If $\alpha=1$ the derivative's magnitude saturates to a constant $1/c$ as $|x|$ grows larger than c, so as a residual increases its effect never decreases but never exceeds a fixed amount. If $\alpha<1$ the derivative's magnitude begins to decrease as $|x|$ grows larger than c (in the language of M-estimation [16], the derivative, aka "influence", is "redescending") so as the residual of an outlier increases, that outlier has *less* effect during gradient descent. The effect of an outlier diminishes as α becomes more negative, and as α approaches $-\infty$ an outlier whose residual magnitude is larger than $3c$ is almost completely ignored.

We can also reason about α in terms of averages. Because the empirical mean of a set of values minimizes total squared error between the mean and the set, and the empirical median similarly minimizes absolute error, minimizing

our loss with $\alpha = 2$ is equivalent to estimating a mean, and with $\alpha = 1$ is similar to estimating a median. Minimizing our loss with $\alpha = -\infty$ is equivalent to local mode-finding [35]. Values of α between these extents can be thought of as smoothly interpolating between these three kinds of averages during estimation.

Our loss function has several useful properties that we will take advantage of. The loss is smooth (*i.e.*, in C^∞) with respect to x, α, and $c > 0$, and is therefore well-suited to gradient-based optimization over its input and its parameters. The loss is zero at the origin, and increases monotonically with respect to $|x|$:

$$\rho\left(0, \alpha, c\right) = 0 \qquad \frac{\partial \rho}{\partial |x|}\left(x, \alpha, c\right) \geq 0 \tag{10}$$

The loss is invariant to a simultaneous scaling of c and x:

$$\forall_{k>0} \;\; \rho(kx, \alpha, kc) = \rho(x, \alpha, c) \tag{11}$$

The loss increases monotonically with respect to α:

$$\frac{\partial \rho}{\partial \alpha}\left(x, \alpha, c\right) \geq 0 \tag{12}$$

This is convenient for graduated non-convexity [4]: we can initialize α such that our loss is convex and then gradually reduce α (and therefore reduce convexity and increase robustness) during optimization, thereby enabling robust estimation that (often) avoids local minima.

We can take the limit of the loss as α approaches infinity, which due to Eq. 12 must be the upper bound of the loss:

$$\rho\left(x, \alpha, c\right) \leq \lim_{\alpha \to +\infty} \rho\left(x, \alpha, c\right) = \exp\left(\frac{1}{2}\left({}^{x}/_{c}\right)^2\right) - 1 \tag{13}$$

We can bound the magnitude of the gradient of the loss, which allows us to better reason about exploding gradients:

$$\left|\frac{\partial \rho}{\partial x}\left(x, \alpha, c\right)\right| \leq \begin{cases} \frac{1}{c}\left(\frac{\alpha-2}{\alpha-1}\right)^{\left(\frac{\alpha-1}{2}\right)} \leq \frac{1}{c} & \text{if } \alpha \leq 1 \\ \frac{|x|}{c^2} & \text{if } \alpha \leq 2 \end{cases} \tag{14}$$

L1 loss is not expressible by our loss, but if c is much smaller than x we can approximate it with $\alpha = 1$:

$$f\left(x, 1, c\right) \approx \frac{|x|}{c} - 1 \quad \text{if } c \ll x \tag{15}$$

See the supplement for other potentially-useful properties that are not used in our experiments.

2. Probability Density Function

With our loss function we can construct a general probability distribution, such that the negative log-likelihood (NLL) of its PDF is a shifted version of our loss function:

$$p\left(x \mid \mu, \alpha, c\right) = \frac{1}{cZ\left(\alpha\right)} \exp\left(-\rho\left(x - \mu, \alpha, c\right)\right) \tag{16}$$

$$Z\left(\alpha\right) = \int_{-\infty}^{\infty} \exp\left(-\rho\left(x, \alpha, 1\right)\right) \tag{17}$$

where $p\left(x \mid \mu, \alpha, c\right)$ is only defined if $\alpha \geq 0$, as $Z\left(\alpha\right)$ is divergent when $\alpha < 0$. For some values of α the partition function is relatively straightforward:

$$\begin{aligned} Z\left(0\right) &= \pi\sqrt{2} & Z\left(1\right) &= 2eK_1(1) \\ Z\left(2\right) &= \sqrt{2\pi} & Z\left(4\right) &= e^{1/4}K_{1/4}(1/4) \end{aligned} \tag{18}$$

where $K_n(\cdot)$ is the modified Bessel function of the second kind. For any rational positive α (excluding a singularity at $\alpha = 2$) where $\alpha = {}^{n}/_{d}$ with $n, d \in \mathbb{N}$, we see that

$$\begin{aligned} Z\left(\frac{n}{d}\right) &= \frac{e^{\left|\frac{2d}{n}-1\right|}\sqrt{\left|\frac{2d}{n}-1\right|}}{(2\pi)^{(d-1)}} G^{0,0}_{p,q}\left(\left.\begin{matrix}\mathbf{a_p} \\ \mathbf{b_q}\end{matrix}\right| \left(\frac{1}{n} - \frac{1}{2d}\right)^{2d}\right) \\ \mathbf{b_q} &= \left\{\left.\frac{i}{n}\right| i = -\frac{1}{2}, ..., n - \frac{3}{2}\right\} \cup \left\{\left.\frac{i}{2d}\right| i = 1, ..., 2d-1\right\} \\ \mathbf{a_p} &= \left\{\left.\frac{i}{n}\right| i = 1, ..., n-1\right\} \end{aligned} \tag{19}$$

where $G(\cdot)$ is the Meijer G-function and $\mathbf{b_q}$ is a multiset (items may occur twice). Because the partition function is difficult to evaluate or differentiate, in our experiments we approximate $\log(Z\left(\alpha\right))$ with a cubic hermite spline (see the supplement for details).

Just as our loss function includes several common loss function as special cases, our distribution includes several common distributions as special cases. When $\alpha = 2$ our distribution becomes a normal (Gaussian) distribution, and when $\alpha = 0$ our distribution becomes a Cauchy distribution. These are also both special cases of Student's t-distribution ($\nu = \infty$ and $\nu = 1$, respectively), though these are the only two points where these two families of distributions intersect. Our distribution resembles the generalized Gaussian distribution [28, 33], except that it is "smoothed" so as to approach a Gaussian distribution near the origin regardless of the shape parameter α. The PDF and NLL of our distribution for different values of α can be seen in Figure 2.

In later experiments we will use the NLL of our general distribution $-\log(p(\cdot|\alpha, c))$ as the loss for training our neural networks, not our general loss $\rho\left(\cdot, \alpha, c\right)$. Critically, using the NLL allows us to treat α as a free parameter, thereby allowing optimization to automatically determine the degree of robustness that should be imposed by the loss being used during training. To understand why the NLL must be used for this, consider a training procedure in which we simply minimize $\rho\left(\cdot, \alpha, c\right)$ with respect to α and our model weights. In this scenario, the monotonicity of our general loss with respect to α (Eq. 12) means that optimization can trivially minimize the cost of outliers by setting α to be as small as possible. Now consider that same training procedure in which we minimize the NLL of our distribution

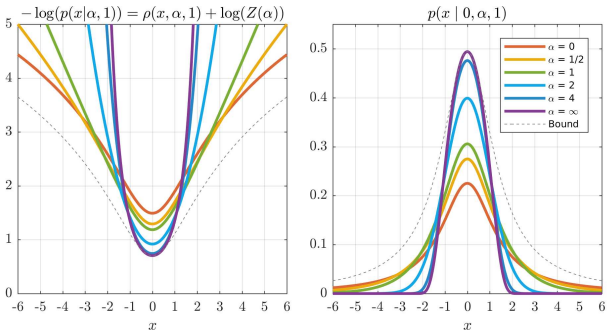

Figure 2. The negative log-likelihoods (left) and probability densities (right) of the distribution corresponding to our loss function when it is defined ($\alpha \geq 0$). NLLs are simply losses (Fig. 1) shifted by a log partition function. Densities are bounded by a scaled Cauchy distribution.

instead of our loss. As can be observed in Figure 2, reducing α will decrease the NLL of outliers but will *increase* the NLL of inliers. During training, optimization will have to choose between reducing α, thereby getting "discount" on large errors at the cost of paying a penalty for small errors, or increasing α, thereby incurring a higher cost for outliers but a lower cost for inliers. This tradeoff forces optimization to judiciously adapt the robustness of the NLL being minimized. As we will demonstrate later, allowing the NLL to adapt in this way can increase performance on a variety of learning tasks, in addition to obviating the need for manually tuning α as a fixed hyperparameter.

Sampling from our distribution is straightforward given the observation that $-\log\left(p\left(x \mid 0, \alpha, 1\right)\right)$ is bounded from below by $\rho(x, 0, 1) + \log(Z(\alpha))$ (shifted Cauchy loss). See Figure 2 for visualizations of this bound when $\alpha = \infty$, which also bounds the NLL for all values of α. This lets us perform rejection sampling using a Cauchy as the proposal distribution. Because our distribution is a location-scale family, we sample from $p\left(x \mid 0, \alpha, 1\right)$ and then scale and shift that sample by c and μ respectively. This sampling approach is efficient, with an acceptance rate between $\sim 45\%$ ($\alpha = \infty$) and 100% ($\alpha = 0$). Pseudocode for sampling is shown in Algorithm 1.

Algorithm 1 Sampling from our general distribution

Input: Parameters for the distribution to sample $\{\mu, \alpha, c\}$
Output: A sample drawn from $p\left(x \mid \mu, \alpha, c\right)$.

while True:
  $x \sim \text{Cauchy}(x_0 = 0, \gamma = \sqrt{2})$
  $u \sim \text{Uniform}(0, 1)$
  if $u < \frac{p(x \mid 0,\alpha,1)}{\exp(-\rho(x,0,1)-\log(Z(\alpha)))}$:
    return $cx + \mu$

3. Experiments

We will now demonstrate the utility of our loss function and distribution with four experiments. None of these results are intended to represent the state-of-the-art for any particular task — our goal is to demonstrate the value of our loss and distribution as useful tools in isolation. 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.

In Sections 3.1 and 3.2 we focus on learning based vision tasks in which training involves minimizing the difference between images: variational autoencoders for image synthesis and self-supervised monocular depth estimation. We will generalize and improve models for both tasks by using our general distribution (either as a conditional distribution in a generative model or by using its NLL as an adaptive loss) and allowing the distribution to *automatically* determine its own degree of robustness. Because robustness is automatic and requires no manually-tuned hyperparameters, we can even allow for the robustness of our loss to be adapted individually for each dimension of our output space — we can have a different degree of robustness at each pixel in an image, for example. As we will show, this approach is particularly effective when combined with image representations such as wavelets, in which we expect to see non-Gaussian, heavy-tailed distributions.

In Sections 3.3 and 3.4 we will build upon existing algorithms for two classic vision tasks (registration and clustering) that both work by minimizing a robust loss that is subsumed by our general loss. We will then replace each algorithm's fixed robust loss with our loss, thereby introducing a continuous tunable robustness parameter α. This generalization allows us to introduce new models in which α is manually tuned or annealed, thereby improving performance. These results demonstrate the value of our loss function when designing classic vision algorithms, by allowing model robustness to be introduced into the algorithm design space as a continuous hyperparameter.

3.1. Variational Autoencoders

Variational autoencoders [22, 30] are a landmark technique for training autoencoders as generative models, which can then be used to draw random samples that resemble training data. We will demonstrate that our general distribution can be used to improve the log-likelihood performance of VAEs for image synthesis on the CelebA dataset [26]. A common design decision for VAEs is to model images using an independent normal distribution on a vector of RGB pixel values [22], and we use this design as our baseline model. Recent work has improved upon this model by using deep, learned, and adversarial loss functions [8, 15, 24]. Though it's possible that our general loss or distribution can add value in these circumstances, to more precisely iso-

late our contribution 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. Again, our goal is not to advance the state of the art for any particular image synthesis task, but is instead to explore the value of our distribution in an experimentally controlled setting.

In our baseline model we give each pixel's normal distribution a variable scale parameter $\sigma^{(i)}$ that will be optimized over during training, thereby allowing the VAE to adjust the scale of its distribution for each output dimension. We can straightforwardly replace this per-pixel normal distribution with a per-pixel general distribution, in which each output dimension is given a distinct shape parameter $\alpha^{(i)}$ in addition to its scale parameter $c^{(i)}$ (*i.e.*, $\sigma^{(i)}$). By letting the $\alpha^{(i)}$ parameters be free variables alongside the scale parameters, training is able to adaptively select both the scale and robustness of the VAE's posterior distribution over pixel values. We restrict all $\alpha^{(i)}$ to be in $(0, 3)$, which allows our distribution to generalize Cauchy ($\alpha = 0$) and Normal ($\alpha = 2$) distributions and anything in between, as well as more platykurtic distributions ($\alpha > 2$) which helps for this task. We limit α to be less than 3 because of the increased risk of numerical instability during training as α increases. We also compare against a Cauchy distribution as an example of a fixed heavy-tailed distribution, and against Student's t-distribution as an example of a distribution that can adjust its own robustness similarly to ours.

Regarding implementation, for each output dimension i we construct unconstrained TensorFlow variables $\{\alpha_\ell^{(i)}\}$ and $\{c_\ell^{(i)}\}$ and define

$$\alpha^{(i)} = (\alpha_{\max} - \alpha_{\min}) \operatorname{sigmoid}\left(\alpha_\ell^{(i)}\right) + \alpha_{\min} \tag{20}$$

$$c^{(i)} = \operatorname{softplus}\left(c_\ell^{(i)}\right) + c_{\min} \tag{21}$$

$$\alpha_{\min} = 0,\ \alpha_{\max} = 3,\ c_{\min} = 10^{-8} \tag{22}$$

The $c_{\min}$ offset avoids degenerate optima where likelihood is maximized by having $c^{(i)}$ approach 0, while $\alpha_{\min}$ and $\alpha_{\max}$ determine the range of values that $\alpha^{(i)}$ can take. Variables are initialized such that initially all $\alpha^{(i)} = 1$ and $c^{(i)} = 0.01$, and are optimized simultaneously with the autoencoder's weights using the same Adam [21] optimizer instance.

Though modeling images using independent distributions on pixel intensities is a popular choice due to its simplicity, classic work in natural image statistics suggest that images are better modeled with heavy-tailed distributions on wavelet-like image decompositions [9, 27]. We therefore train additional models in which our decoded RGB per-pixel images are linearly transformed into spaces that better model natural images before computing the NLL of our distribution. For this we use the DCT [1] and the CDF 9/7 wavelet decomposition [7], both with a YUV colorspace. These representations resemble the JPEG and JPEG 2000 compression standards, respectively.

	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

Table 1. Validation set ELBOs (higher is better) for our variational autoencoders. Models using our general distribution better maximize the likelihood of unseen data than those using normal or Cauchy distributions (both special cases of our model) for all three image representations, and perform similarly to Student's t-distribution (a different generalization of normal and Cauchy distributions). The best and second best performing techniques for each representation are colored orange and yellow respectively.

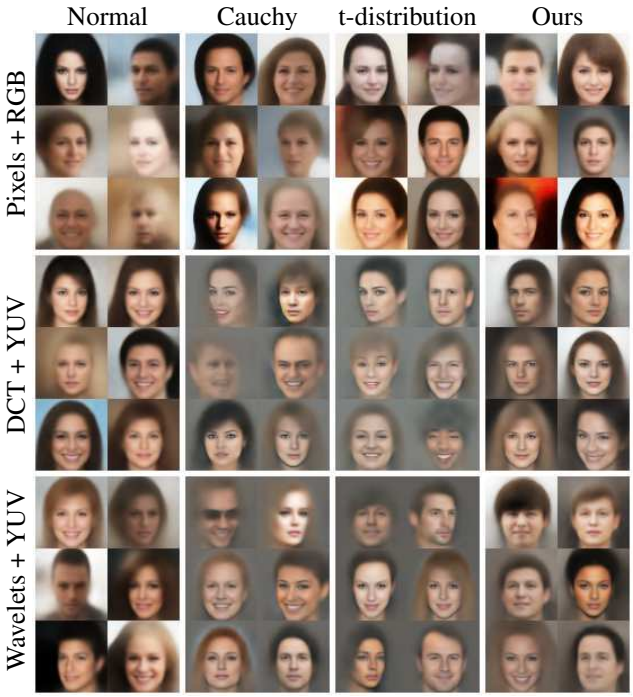

Figure 3. Random samples from our variational autoencoders. We use either normal, Cauchy, Student's t, or our general distributions (columns) to model the coefficients of three different image representations (rows). Because our distribution can adaptively interpolate between Cauchy-like or normal-like behavior for each coefficient individually, using it results in sharper and higher-quality samples (particularly when using DCT or wavelet representations) and does a better job of capturing low-frequency image content than Student's t-distribution.

Our results can be seen in Table 1, where we report the validation set evidence lower bound (ELBO) for all combinations of our four distributions and three image representations, and in Figure 3, where we visualize samples from these models. We see that our general distribution per-

forms similarly to a Student's t-distribution, with both producing higher ELBOs than any fixed distribution across all representations. These two adaptive distributions appear to have complementary strengths: ours can be more platykurtic ($\alpha > 2$) while a t-distribution can be more leptokurtic ($\nu < 1$), which may explain why neither model consistently outperforms the other across representations. Note that the t-distribution's NLL does not generalize the Charbonnier, L1, Geman-McClure, or Welsch losses, so unlike ours it will not generalize the losses used in the other tasks we will address. For all representations, VAEs trained with our general distribution produce sharper and more detailed samples than those trained with normal distributions. Models trained with Cauchy and t-distributions preserve high-frequency detail and work well on pixel representations, but systematically fail to synthesize low-frequency image content when given non-pixel representations, as evidenced by the gray backgrounds of those samples. Comparing performance across image representations shows that the "Wavelets + YUV" representation best maximizes validation set ELBO — though if we were to limit our model to only normal distributions the "DCT + YUV" model would appear superior, suggesting that there is value in reasoning jointly about distributions and image representations. After training we see shape parameters $\{\alpha^{(i)}\}$ that span $(0, 2.5)$, suggesting that an adaptive mixture of normal-like and Cauchy-like distributions is useful in modeling natural images, as has been observed previously [29]. Note that this adaptive robustness is just a consequence of allowing $\{\alpha_\ell^{(i)}\}$ to be free variables during training, and requires no manual parameter tuning. See the supplement for more samples and reconstructions from these models, and a review of our experimental procedure.

3.2. Unsupervised Monocular Depth Estimation

Due to the difficulty of acquiring ground-truth direct depth observations, there has been recent interest in "unsupervised" monocular depth estimation, in which stereo pairs and geometric constraints are used to directly train a neural network [10, 11, 14, 41]. We use [41] as a representative model from this literature, which is notable for its estimation of depth *and* camera pose. This model is trained by minimizing the differences between two images in a stereo pair, where one image has been warped to match the other according to the depth and pose predictions of a neural network. In [41] that difference between images is defined as the absolute difference between RGB values. We will replace that loss with different varieties of our general loss, and demonstrate that using annealed or adaptive forms of our loss can improve performance.

The absolute loss in [41] 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 α to vary. Following our observation from Section 3.1 that YUV wavelet representations work well when modeling images with our loss function, we impose our loss on a YUV wavelet decomposition instead of the RGB pixel representation of [41]. The only changes we made to the code from [41] were to replace its loss function with our own and to remove the model components that stopped yielding any improvement after the loss function was replaced (see the supplement for details). All training and evaluation was performed on the KITTI dataset [12] using the same training/test split as [41].

Results can be seen in Table 2. We present the error and accuracy metrics used in [41] and our own "average" error measure: the geometric mean of the four errors and one minus the three accuracies. The "Baseline" models use the loss function of [41], and we present both the numbers in [41] ("as reported") and our own numbers from running

	lower is better					higher is better		
	Avg	AbsRel	SqRel	RMS	logRMS	<1.25	$<1.25^2$	$<1.25^3$
Baseline [41] as reported	0.407	0.221	2.226	7.527	0.294	0.676	0.885	0.954
Baseline [41] 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 $\alpha = 2 \rightarrow 0$	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

Table 2. Results on unsupervised monocular depth estimation using the KITTI dataset [12], building upon the model from [41] ("Baseline"). By replacing the per-pixel loss used by [41] with several variants of our own per-wavelet general loss function in which our loss's shape parameters are fixed, annealed, or adaptive, we see a significant performance improvement. The top three techniques are colored red, orange, and yellow for each metric.

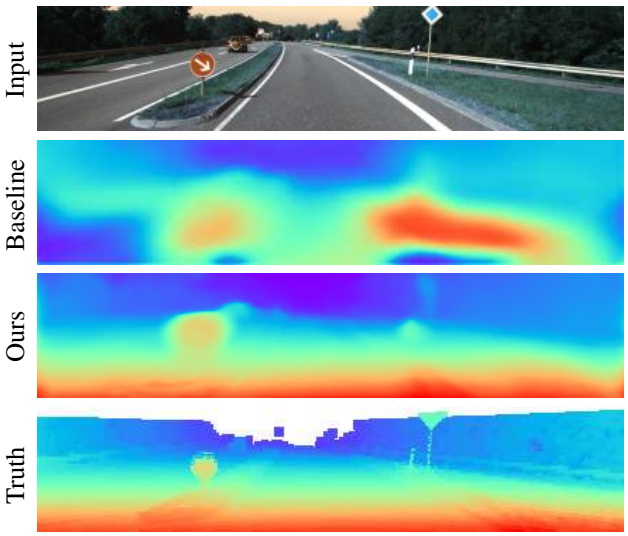

Figure 4. Monocular depth estimation results on the KITTI benchmark using the "Baseline" network of [41]. Replacing only the network's loss function with our "adaptive" loss over wavelet coefficients results in significantly improved depth estimates.

the code from [41] ourselves ("reproduced"). The "Ours" entries all use our general loss imposed on wavelet coefficients, but for each entry we use a different strategy for setting the shape parameter or parameters. We keep our loss's scale c fixed to 0.01, thereby matching the fixed scale assumption of the baseline model and roughly matching the shape of its L1 loss (Eq. 15). To avoid exploding gradients we multiply the loss being minimized by c, thereby bounding gradient magnitudes by residual magnitudes (Eq. 14). For the "fixed" models we use a constant value for α for all wavelet coefficients, and observe that though performance is improved relative to the baseline, no single value of α is optimal. The $\alpha = 1$ entry is simply a smoothed version of the L1 loss used by the baseline model, suggesting that just using a wavelet representation improves performance. In the "annealing $\alpha = 2 \rightarrow 0$" model we linearly interpolate α from 2 (L2) to 0 (Cauchy) as a function of training iteration, which outperforms all "fixed" models. In the "adaptive $\alpha \in (0, 2)$" model we assign each wavelet coefficient its own shape parameter as a free variable and we allow those variables to be optimized alongside our network weights during training as was done in Section 3.1, but with $\alpha_{\min} = 0$ and $\alpha_{\max} = 2$. This "adaptive" strategy outperforms the "annealing" and all "fixed" strategies, thereby demonstrating the value of allowing the model to adaptively determine the robustness of its loss during training. Note that though the "fixed" and "annealed" strategies only require our general loss, the "adaptive" strategy requires that we use the NLL of our general distribution as our loss — otherwise training would simply drive α to be as small as possible due to the monotonicity of our loss with respect to α, causing performance to degrade to the "fixed $\alpha = 0$" model. Comparing the "adaptive" model's performance to that of the "fixed" models suggests that, as in Section 3.1, no single setting of α is optimal for all wavelet coefficients. Overall, we see that just replacing the loss function of [41] with our adaptive loss on wavelet coefficients reduces average error by $\sim 17\%$.

In Figure 4 we compare our "adaptive" model's output to the baseline model and the ground-truth depth, and demonstrate a substantial qualitative improvement. See the supplement for many more results, and for visualizations of the per-coefficient robustness selected by our model.

3.3. Fast Global Registration

Robustness is often a core component of geometric registration [37]. The Fast Global Registration (FGR) algorithm of [40] finds the rigid transformation $\mathbf{T}$ that aligns point sets $\{\mathbf{p}\}$ and $\{\mathbf{q}\}$ by minimizing the following loss:

$$\sum_{(\mathbf{p},\mathbf{q})} \rho_{gm}\left(\|\mathbf{p} - \mathbf{T}\mathbf{q}\|, c\right) \tag{23}$$

	Mean RMSE ×100			Max RMSE ×100		
$\sigma =$	0	0.0025	0.005	0	0.0025	0.005
FGR [40]	0.373	0.518	0.821	0.591	1.040	1.767
shape-annealed gFGR	0.374	0.510	**0.802**	0.590	0.997	1.670
gFGR*	**0.370**	**0.509**	0.806	**0.545**	**0.961**	**1.669**

Table 3. Results on the registration task of [40], in which we compare their "FGR" algorithm to two versions of our "gFGR" generalization.

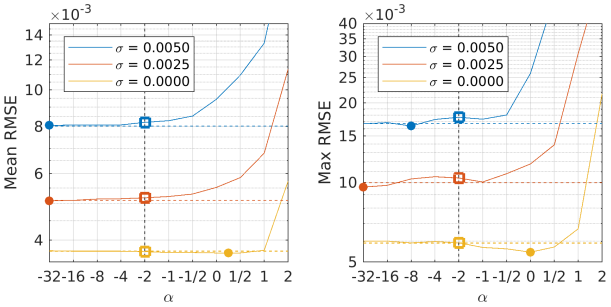

Figure 5. Performance (lower is better) of our gFGR algorithm on the task of [40] as we vary our shape parameter α, with the lowest-error point indicated by a circle. FGR (equivalent to gFGR with $\alpha = -2$) is shown as a dashed line and a square, and shape-annealed gFGR for each noise level is shown as a dotted line.

where $\rho_{gm}(\cdot)$ is Geman-McClure loss. By using the Black and Rangarajan duality between robust estimation and line processes [3] FGR is capable of producing high-quality registrations at high speeds. Because Geman-McClure loss is a special case of our loss, and because we can formulate our loss as an outlier process (see supplement), we can generalize FGR to an arbitrary shape parameter α by replacing $\rho_{gm}(\cdot, c)$ with our $\rho(\cdot, \alpha, c)$ (where setting $\alpha = -2$ reproduces FGR).

This generalized FGR (gFGR) enables algorithmic improvements. FGR iteratively solves a linear system while annealing its scale parameter c, which has the effect of gradually introducing nonconvexity. gFGR enables an alternative strategy in which we directly manipulate convexity by annealing α instead of c. This "shape-annealed gFGR" follows the same procedure as [40]: 64 iterations in which a parameter is annealed every 4 iterations. Instead of annealing c, we set it to its terminal value and instead anneal α over the following values:

$$2, 1, 1/2, 1/4, 0, -1/4, -1/2, -1, -2, -4, -8, -16, -32$$

Table 3 shows results for the 3D point cloud registration task of [40] (Table 1 in that paper), which shows that annealing shape produces moderately improved performance over FGR for high-noise inputs, and behaves equivalently in low-noise inputs. This suggests that performing graduated non-convexity by directly adjusting a shape parameter that controls non-convexity — a procedure that is enabled by our general loss – is preferable to indirectly controlling non-convexity by annealing a scale parameter.

Dataset	AC-W	N-Cuts [32]	LDMGI [36]	PIC [38]	RCC-DR [31]	RCC [31]	gRCC*	Rel. Impr.
YaleB	0.767	0.928	0.945	0.941	0.974	0.975	**0.975**	0.4%
COIL-100	0.853	0.871	0.888	**0.965**	0.957	0.957	0.962	11.6%
MNIST	0.679	-	0.761	-	0.828	0.893	**0.901**	7.9%
YTF	0.801	0.752	0.518	0.676	0.874	0.836	**0.888**	31.9%
Pendigits	0.728	0.813	0.775	0.467	0.854	0.848	**0.871**	15.1%
Mice Protein	0.525	0.536	0.527	0.394	0.638	0.649	**0.650**	0.2%
Reuters	0.471	0.545	0.523	0.057	0.553	0.556	**0.561**	1.1%
Shuttle	0.291	0.000	**0.591**	-	0.513	0.488	0.493	0.9%
RCV1	0.364	0.140	0.382	0.015	**0.442**	0.138	0.338	23.2%

Table 4. Results on the clustering task of [31] where we compare their "RCC" algorithm to our "gRCC*" generalization in terms of AMI on several datasets. We also report the AMI increase of "gRCC*" with respect to "RCC". Baselines are taken from [31].

Another generalization is to continue using the c-annealing strategy of [40], but treat α as a hyperparameter and tune it independently for each noise level in this task. In Figure 5 we set α to a wide range of values and report errors for each setting, using the same evaluation of [40]. We see that for high-noise inputs more negative values of α are preferable, but for low-noise inputs values closer to 0 are optimal. We report the lowest-error entry for each noise level as "gFGR*" in Table 3 where we see a significant reduction in error, thereby demonstrating the improvement that can be achieved from treating robustness as a hyperparameter.

3.4. Robust Continuous Clustering

In [31] robust losses are used for unsupervised clustering, by minimizing:

$$\sum_i \|\mathbf{x}_i - \mathbf{u}_i\|_2^2 + \lambda \sum_{(p,q)\in\mathcal{E}} w_{p,q}\rho_{gm}\left(\|\mathbf{u}_p - \mathbf{u}_q\|_2\right) \quad (24)$$

where $\{\mathbf{x}_i\}$ is a set of input datapoints, $\{\mathbf{u}_i\}$ is a set of "representatives" (cluster centers), and $\mathcal{E}$ is a mutual k-nearest neighbors (m-kNN) graph. As in Section 3.3, $\rho_{gm}(\cdot)$ is Geman-McClure loss, which means that our loss can be used to generalize this algorithm. Using the RCC code provided by the authors (and keeping all hyperparameters fixed to their default values) we replace Geman-McClure loss with our general loss and then sweep over values of α. In Figure 6 we show the adjusted mutual information (AMI, the metric used by [31]) of the resulting clustering for each value of α on the datasets used in [31], and in Table 4 we report the AMI for the best-performing value of α for each dataset as "gRCC*". On some datasets performance is insensitive to α, but on others adjusting α improves performance by as much as 32%. This improvement demonstrates the gains that can be achieved by introducing robustness as a hyperparameter and tuning it accordingly.

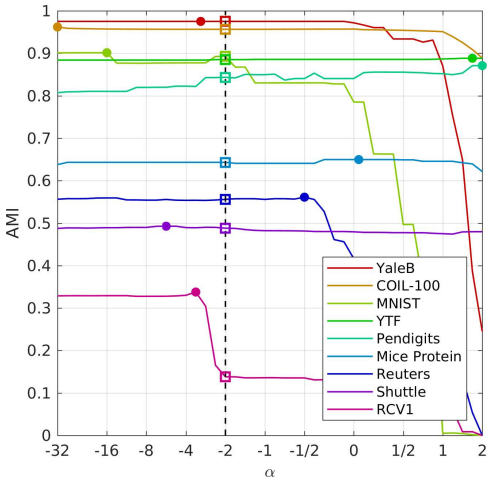

Figure 6. Performance (higher is better) of our gRCC algorithm on the clustering task of [31], for different values of our shape parameter α, with the highest-accuracy point indicated by a dot. Because the baseline RCC algorithm is equivalent to gRCC with $\alpha = -2$, we highlight that α value with a dashed line and a square.

4. Conclusion

We have presented a two-parameter loss function that generalizes many existing one-parameter robust loss functions: the Cauchy/Lorentzian, Geman-McClure, Welsch/Leclerc, generalized Charbonnier, Charbonnier/pseudo-Huber/L1-L2, and L2 loss functions. By reducing a family of discrete single-parameter losses to a single function with two continuous parameters, our loss enables the convenient exploration and comparison of different robust penalties. This allows us to generalize and improve algorithms designed around the minimization of some fixed robust loss function, which we have demonstrated for registration and clustering. When used as a negative log-likelihood, this loss gives a general probability distribution that includes normal and Cauchy distributions as special cases. This distribution lets us train neural networks in which the loss has an adaptive degree of robustness for each output dimension, which allows training to automatically determine how much robustness should be imposed by the loss without any manual parameter tuning. When this adaptive loss is paired with image representations in which variable degrees of heavy-tailed behavior occurs, such as wavelets, this adaptive training approach allows us to improve the performance of variational autoencoders for image synthesis and of neural networks for unsupervised monocular depth estimation.

Acknowledgements: Thanks to Rob Anderson, Jesse Engel, David Gallup, Ross Girshick, Jaesik Park, Ben Poole, Vivek Rathod, and Tinghui Zhou.

References

[1] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE Transactions on Computers*, 1974.

[2] Michael J Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *CVIU*, 1996.

[3] Michael J. Black and Anand Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *IJCV*, 1996.

[4] Andrew Blake and Andrew Zisserman. *Visual Reconstruction*. MIT Press, 1987.

[5] Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. *ICIP*, 1994.

[6] Qifeng Chen and Vladlen Koltun. Fast mrf optimization with application to depth reconstruction. *CVPR*, 2014.

[7] Albert Cohen, Ingrid Daubechies, and J-C Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 1992.

[8] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *NIPS*, 2016.

[9] David J. Field. Relations between the statistics of natural images and the response properties of cortical cells. *JOSA A*, 1987.

[10] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. *CVPR*, 2016.

[11] Ravi Garg, BG Vijay Kumar, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. *ECCV*, 2016.

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. *CVPR*, 2012.

[13] Stuart Geman and Donald E. McClure. Bayesian image analysis: An application to single photon emission tomography. *Proceedings of the American Statistical Association*, 1985.

[14] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *CVPR*, 2017.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NIPS*, 2014.

[16] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley, 1986.

[17] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman and Hall/CRC, 2015.

[18] Peter J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 1964.

[19] Peter J. Huber. *Robust Statistics*. Wiley, 1981.

[20] John E. Dennis Jr. and Roy E. Welsch. Techniques for nonlinear least squares and robust regression. *Communications in Statistics-simulation and Computation*, 1978.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[22] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[23] Philipp Krähenbühl and Vladlen Koltun. Efficient nonlocal regularization for optical flow. *ECCV*, 2012.

[24] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *ICML*, 2016.

[25] Yvan G Leclerc. Constructing simple stable descriptions for image partitioning. *IJCV*, 1989.

[26] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *ICCV*, 2015.

[27] Stéphane Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *TPAMI*, 1989.

[28] Saralees Nadarajah. A generalized normal distribution. *Journal of Applied Statistics*, 2005.

[29] Javier Portilla, Vasily Strela, Martin J. Wainwright, and Eero P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE TIP*, 2003.

[30] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *ICML*, 2014.

[31] Sohil Atul Shah and Vladlen Koltun. Robust continuous clustering. *PNAS*, 2017.

[32] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *TPAMI*, 2000.

[33] M Th Subbotin. On the law of frequency of error. *Matematicheskii Sbornik*, 1923.

[34] Deqing Sun, Stefan Roth, and Michael J. Black. Secrets of optical flow estimation and their principles. *CVPR*, 2010.

[35] Rein van den Boomgaard and Joost van de Weijer. On the equivalence of local-mode finding, robust estimation and mean-shift analysis as used in early vision tasks. *ICPR*, 2002.

[36] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. Image clustering using local discriminant models and global integration. *TIP*, 2010.

[37] Christopher Zach. Robust bundle adjustment revisited. *ECCV*, 2014.

[38] Wei Zhang, Deli Zhao, and Xiaogang Wang. Agglomerative clustering via maximum incremental path integral. *Pattern Recognition*, 2013.

[39] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting, 1995.

[40] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. *ECCV*, 2016.

[41] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. *CVPR*, 2017.