STABILIZED TRAINING OF JOINT ENERGY-BASED MODELS AND ITS PRACTICAL APPLICATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

The recently proposed Joint Energy-based Model (JEM) interprets discriminatively trained classifier $p(y|x)$ as an energy model, which is also trained as a generative model describing the distribution of the input observations $p(x)$. The JEM training relies on "positive examples" (i.e. examples from the training data set) as well as on "negative examples", which are samples from the modeled distribution $p(x)$ generated by means of Stochastic Gradient Langevin Dynamics (SGLD). Unfortunately, SGLD often fails to deliver negative samples of sufficient quality during the standard JEM training, which causes a very unbalanced contribution from the positive and negative examples when calculating gradients for JEM updates. As a consequence, the standard JEM training is quite unstable requiring careful tuning of hyper-parameters and frequent restarts when the training starts diverging. This makes it difficult to apply JEM to different neural network architectures, modalities, and tasks. In this work, we propose a training procedure that stabilizes SGLD-based JEM training (ST-JEM) by balancing the contribution from the positive and negative examples. We also propose to add an additional "regularization" term to the training objective – MI between the input observations x and output labels y – which encourages the JEM classifier to make more certain decisions about output labels. We demonstrate the effectiveness of our approach on the CIFAR10 and CIFAR100 tasks. We also consider the task of classifying phonemes in a speech signal, for which we were not able to train JEM without the proposed stabilization. We show that a convincing speech can be generated from the trained model. Alternatively, corrupted speech can be de-noised by bringing it closer to the modeled speech distribution using a few SGLD iterations. We also propose and discuss additional applications of the trained model.

1 INTRODUCTION

One of the most common Machine Learning tasks is to classify input data points into chosen categories which typically is accomplished by a discriminatively trained classifier. Having an enormous amount of data and a powerful machine learning model of suitable architecture with a huge number of learnable parameters is usually enough to get close to state-of-the-art performance. The alternative approach of training a generative model and then inferring the posterior probability over the possible categories can outperform discriminative models only in the low-resource settings and usually fails to be competitive due to its restricted modeling power otherwise. The utility of explicit generative models then lies in the access to the likelihood of the input data useful for e.g. detecting outliers while implicit generative models are evaluated based on the capability to generate realistic and diverse input, especially when applied to images. The promise of generative models to avoid expensive labeling of unlabeled data (for which a discriminative classifier has no use) is shadowed by the recent success of self-supervised techniques that take advantage of self-contained information in the time sequence or leverage known input data manipulations (e.g. shift and resize of images) to introduce labels used to train a discriminative model. Self-supervised models are usually used as pre-trained models to extract embeddings that work well in the downstream task.

Recently, Grathwohl et al. (2019) showed that every discriminatively trained classifier can be seen as an energy-based model modeling the joint distribution between the input data x and the label y. In fact, when the discriminator is trained in a standard way, we are only training the model to provide us with a good estimate of $p(y|x)$ while $p(x)$ is not being optimized at all. In order to optimize

$p(x)$, authors used Stochastic Gradient Langevin Dynamics (SGLD) as it was described in Welling & Teh (2011) to sample[1] from the modeled distribution and called the resulting model Joint Energy-based Model (JEM). Authors demonstrated that the longer training time and slight performance degradation (compared to its strictly discriminative counterpart) is compensated by the possibility to generate either category-conditional or unconditional samples, robustness against adversarial attacks, improved calibration, and out-of-distribution detection. Unfortunately, it is difficult to take these models and easily apply them to new tasks as the training often diverges, and restarting the training from the last saved epoch seems to be the only reliable solution to reach a stable optimum. This prevents the community from conducting a deeper exploration of these models. As a response, VERA (Duvenaud et al. (2021)) introduced a stable way of training JEMs. In this work, the authors demonstrate that they are able to speed up the training by introducing an auxiliary model (generator) and they are capable of producing high-quality images based on that generator.

2 OUR CONTRIBUTION

In this work, we propose a method to train JEM without the use of any auxiliary model whose training does not diverge even though we use SGLD samples. We propose to add a term that maximizes mutual information (MI) between the inputs x and labels y into the previously defined objective function and approximate it for necessary simplification. This reveals an alternative way of training and allows us to reach superior accuracy compared to results reported by Grathwohl et al. (2019) using JEMs on the CIFAR-10 and CIFAR-100 dataset while being able to generate images of similar quality.

Our method allows for training even in the case of not using any SGLD samples. In fact, this can be interpreted as optimizing JEM with the domain defined only over the training data points. We cannot sample from the model as we assume that outside of the domain of train data, $p(x)$ is strictly 0 and the model output is ignored.. Over this restricted domain, we don't need to approximate MI but we can optimize it directly. This results in a boost in performance, explaining why our JEM model outperforms the baseline which was not the the case for the original training of JEM.

Maximizing approximate mutual information term added to the loss function helped us realize the stabilization trick, but we show that the same trick can also be directly applied to the joint distribution $p(x, y)$ (Excluding MI) reaching very similar performance.

Grathwohl et al. (2019) demonstrated that JEM can provide better calibration in the low-resource setting. They used only a portion of the original labeled data and treated the rest of the data as unlabeled. Unfortunately, they reported that adding unlabeled data had no effect on the accuracy. We show that low-resource or more difficult problems are the ones where we see the largest boost in accuracy and calibration compared to a simple discriminative model. Moreover, we extended our approach to be able to handle unlabeled data and we actually do report a boost in accuracy.

The updated version of SGLD has fixed hyperparameters. By carefully monitoring the SGLD process during the training, we noticed that the optimal hyper-parameters change during the training. Using the same hyper-parameters can eventually lead to the state when the updated version of SGLD does not provide any reasonable samples. In our approach, SGLD samples that are not competitive, do not influence the stability of the training. It opens the possibility to do a simple exploration of the different hyper-parameters, resulting in competitive samples again. This is a promising way to improve the speed and the quality of generated samples and more sophisticated ways should be explored in the future. We also found out that the quality and the speed of generated samples can be greatly affected by these and for conditional sampling, each class might have different optimal hyper-parameters.

Our main motivation to stabilize JEM training is to be able to apply it to a different modality (speech). We train JEM to model $p(x, y)$ of the input frame and its context[2] x and the phoneme label of the central frame. We discuss the potential future use of JEM as a single model capable of being ASR, TTS, denoiser, speaker recognizer, voice conversion, the source separator or inpainting (also inpainting conditional on the category). The generative part of JEM can further be leveraged

[1] These samples are sometimes called negative samples as opposed to train data being called positive samples.

[2] We used 80 log Mel-filter banks as a frame representation.

when estimating uncertainty [ignore the uncertain part of the input for ASR, SPK-ID] or when combining the output of different models. Furthermore, we demonstrate that we are able to generate interesting conditional and unconditional speech, and show promising preliminary results on denoising and model combination.

Our approach allows us to include any number of samples from SGLD and we typically use 8 samples per each batch of 64[3] training examples. Increasing the number of samples increases the quality of generated images but at the same time slightly degrades the performance of a classifier. We have also observed an increase in the quality of generated images (CIFAR-10, CIFAR-100) during the training usually corresponds to a decrease in accuracy, this suggests that the model might not be large enough as we haven't noticed the same behavior when training on the speech datasets. An alternative way of speeding up the training is generating more SGLD samples but doing so only once per few batches. We have observed that increasing the number of SGLD steps can results in a much better quality of generated images in both training and inference but it significantly slows down the training. [When finished, add references from this section to the parts in the body/appendix where it is described with more details]

3 BACKGROUND

In this section, we provide an overview and explanation of previously introduced techniques that helps to follow our reasoning. More detailed explanations of these techniques can be found in referenced literature.

3.1 ENERGY-BASED MODELS

Energy-based models can represent a complex probability distribution. This is a special case of distributions because there is no simple way to sample from such a distribution - we can only evaluate probability distribution up to an unknown normalizer. Probability distribution over a continuous variable $\mathbf{x}$ is then defined as

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{e^{-E_{\boldsymbol{\theta}}(\mathbf{x})}}{Z_{\boldsymbol{\theta}}} = \frac{e^{f^x_{\boldsymbol{\theta}}(\mathbf{x})}}{Z_{\boldsymbol{\theta}}}, \tag{1}$$

where $E_{\boldsymbol{\theta}}(\mathbf{x})$ is the energy function that assigns a score to each continuous input $\mathbf{x} \in \mathbf{X}, \mathbf{X} = \mathbb{R}^{D_x}$. To make sure that $p_{\boldsymbol{\theta}}(\mathbf{x})$ is a properly normalized distribution, we define a partition function as $Z_{\boldsymbol{\theta}} = \int_{\mathbf{x}} e^{f^x_{\boldsymbol{\theta}}(\mathbf{x})} d\mathbf{x}$. We can further modify Equation 1 to define the joint distribution $p_{\boldsymbol{\theta}}(\mathbf{x}, y)$ of both continuous input $\mathbf{x}$ and a discrete label $y \in \mathrm{Y}, \mathrm{Y} = \mathbb{N}^+, y \leq D_y$ as

$$p_{\boldsymbol{\theta}}(\mathbf{x}, y = i) = \frac{e^{-E_{\boldsymbol{\theta}}(\mathbf{x}, y=i)}}{Z_{\boldsymbol{\theta}}} = \frac{e^{f_{\boldsymbol{\theta}}(\mathbf{x}, y=i)}}{Z_{\boldsymbol{\theta}}} = \frac{e^{f_{\boldsymbol{\theta}}(\mathbf{x})_i}}{Z_{\boldsymbol{\theta}}} = \frac{e^{q_i}}{Z_{\boldsymbol{\theta}}} \tag{2}$$

A typical way of obtaining the value of negative energy $-E_{\boldsymbol{\theta}}(\mathbf{x}, y)$ is via a function $f_{\boldsymbol{\theta}}(\mathbf{x}, y) : \mathbf{x}, y \mapsto -E_{\boldsymbol{\theta}}(\mathbf{x}, y)$, but we focus on an alternative way by using a vector-valued function $f_{\boldsymbol{\theta}}(\mathbf{x}) : \mathbf{x} \mapsto \mathbf{q}$, $\mathbf{q} \in \mathbb{R}^{D_y}$, where the i-th element of the vector $\mathbf{q}$ (denoted as q_i) represents $-E_{\boldsymbol{\theta}}(\mathbf{x}, y = i)$. In both cases, the partition function is given by $Z_{\boldsymbol{\theta}} = \sum_y \int_{\mathbf{x}} e^{-E_{\boldsymbol{\theta}}(\mathbf{x}, y)} d\mathbf{x}$. Notice that EBMs do not provide an access to likelihood values because $Z_{\boldsymbol{\theta}}$ is intractable. Maximizing the log-likelihood of one data point $\mathbf{x}$ with respect to the parameters $\boldsymbol{\theta}$ is not straightforward either. In order to compute the gradient, we need to evaluate the intractable expectation over $\mathbf{x}$ as shown in Equation 3.

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = \nabla_{\boldsymbol{\theta}} f^x_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}} f^x_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbb{E}_{(\tilde{\mathbf{x}}) \sim p_{\boldsymbol{\theta}}(\mathbf{x})} \left[\nabla_{\boldsymbol{\theta}} f^x_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) \right] \tag{3}$$

Likewise, expressing $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, y)$ leads to the same conclusion (Equation 4).

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x}, y) = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y - \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y - \mathbb{E}_{(\tilde{\mathbf{x}}, j) \sim p_{\boldsymbol{\theta}}(\mathbf{x}, y)} \left[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\tilde{\mathbf{x}})_j \right] \tag{4}$$

One popular strategy to approximate these expectations is to generate samples from the modeled distribution, the ones based on Markov chain Monte Carlo (MCMC) are of particular interest to us, specifically, Stochastic Gradient Langevin Dynamics (SGLD) Welling & Teh (2011)[4]. SGLD is

[3] 100 for speech experiments

[4] An overview of alternative approaches to train EBMs is presented in Song & Kingma (2021).

capable of generating samples from $p_{\boldsymbol{\theta}}(\mathbf{x})$ induced by a neural network $f^x_{\boldsymbol{\theta}}(\mathbf{x})$ through an iterative procedure

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \frac{\alpha^t}{2}\nabla_{\mathbf{x}} f^x_{\boldsymbol{\theta}}(\mathbf{x}^t) + \mathbf{u}^t, \qquad u^t_i \sim \mathcal{N}(0, \alpha^t), \;\; 1 \leq i \leq D_x, \tag{5}$$

starting from a random input $\mathbf{x}^0$. In theory, when $\sum_t \alpha^t = \infty$ and $\sum_t (\alpha^t)^2 < \infty$, it is guaranteed that $\mathbf{x}^t$ becomes a sample from $p_{\boldsymbol{\theta}}(\mathbf{x})$ as $t \to \infty$. In practice, we must resort to an updated (sped up) version of SGLD by significantly limiting the number of steps. The number of steps needed to generate a reasonable sample can be even smaller when the initial sample $\mathbf{x}^0$ is chosen carefully – one of the common techniques is called Persistent Contrastive Divergence (PCD), where the buffer of previously generated samples is maintained and initial samples are drawn from the buffer most of the time as described in Du & Mordatch (2019). Another trick is to reduce the amount of noise $\mathbf{u}^t$ added at each step. Notice, that in order to obtain a sample from $p_{\boldsymbol{\theta}}(\mathbf{x})$ when modeling the joint distribution $p_{\boldsymbol{\theta}}(\mathbf{x}, y)$, we need to replace $f^x_{\boldsymbol{\theta}}(\mathbf{x})$ by $\log \sum_i e^{f_{\boldsymbol{\theta}}(\mathbf{x})_i}$ in Equation 5. If we desire to get a sample having a particular label $y = i$[5], $f^x_{\boldsymbol{\theta}}(\mathbf{x})$ is replaced by $f_{\boldsymbol{\theta}}(\mathbf{x})_i$.

3.2 Classification

Training a classifier parameterized by $\boldsymbol{\theta}$ is achieved by minimizing the cross-entropy H between the true posterior distribution p and modelled posterior distribution $p_{\boldsymbol{\theta}}$ over the possible values of the label y. For a given single input data point $\mathbf{x}$, we calculate

$$H(p, p_{\boldsymbol{\theta}}) = -\sum_{y \in \mathrm{Y}} p(y \mid \mathbf{x}) \log p_{\boldsymbol{\theta}}(y \mid \mathbf{x}). \tag{6}$$

We can directly access the true posterior distribution $p(y \mid \mathbf{x})$ but $p(x)$ is inaccessible, for that reason we approximate the expectation over the true data distribution $\mathbb{E}_{p(\mathbf{x})}\left[\mathrm{H}(p, p_{\boldsymbol{\theta}})\right]$ by an empirical distribution p_{data} that is provided in form of tuples $(\mathbf{x^i}, y^i) \in \mathrm{DS}$, where $p_{data}(y = y^i \mid \mathbf{x} = \mathbf{x^i}) = 1$, therefore

$$\mathbb{E}_{p_{\mathbf{x}}}\left[\mathrm{H}(p, p_{\boldsymbol{\theta}})\right] \approx \underbrace{\mathbb{E}_{\mathbf{x} \sim p_{data}(x)} \mathbb{E}_{y \sim p(y|\mathbf{x})}}_{\mathbb{E}_{(\mathbf{x},y) \sim p_{data}(\mathbf{x},y)}} [-\log p_{\boldsymbol{\theta}}(y \mid \mathbf{x})] = \frac{1}{|\mathrm{DS}|} \sum_{i=0}^{|\mathrm{DS}|} -\log p_{\boldsymbol{\theta}}(y^i \mid \mathbf{x^i}) \tag{7}$$

In order to model $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$, it is common to use Softmax function $\mathrm{SM}(\mathbf{z}) : \mathbb{R}^{D_y} \to (0, 1)^{D_y}$ (Equation 8 that transforms logits $\mathbf{z} = g_{\boldsymbol{\theta}}(\mathbf{x})$ into the vector of posterior probabilities, where $g_{\boldsymbol{\theta}}(\mathbf{x})$ is usually a neural network.

$$p_{\boldsymbol{\theta}}(y = i \mid \mathbf{x}) = \mathrm{SM}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{D_y} e^{z_j}} \tag{8}$$

Softmax has one extra degree of freedom, meaning that for each $\mathbf{x}$, we can add any value $c(\mathbf{x})$ to logits $\mathbf{z}$ and it has no effect on the posterior distribution. Notice that for chosen $\mathbf{x}$, this value must be the same for all y, therefore we can obtain this value by any function $c(\mathbf{x}) : \mathbb{R}^{D_x} \to \mathbb{R}$:

$$\mathrm{SM}_i(\mathbf{z} + c(\mathbf{x})) = \frac{e^{z_i + c(\mathbf{x})}}{\sum_{j=1}^{D_y} e^{z_j + c(\mathbf{x})}} = \frac{e^{c(\mathbf{x})} e^{z_i}}{e^{c(\mathbf{x})} \sum_{j=1}^{D_y} e^{z_j}} = \mathrm{SM}_i(\mathbf{z}) \tag{9}$$

From Equations 8 and 9 (replacing $e^{c(\mathbf{x})}$ by $k(\mathbf{x})$ for brevity), we can see:

$$p_{\boldsymbol{\theta}}(y = i \mid \mathbf{x}) = \frac{e^{c(\mathbf{x})} e^{z_i}}{e^{c(\mathbf{x})} \sum_{j=1}^{D_y} e^{z_j}} = \frac{k(\mathbf{x}) e^{z_i}}{k(\mathbf{x}) \sum_{j=1}^{D_y} e^{z_j}} = \frac{\frac{e^{z_i}}{k(\mathbf{x})}}{\sum_{j=1}^{D_y} \frac{e^{z_j}}{k(\mathbf{x})}} \tag{10}$$

3.3 Energy-based Classifier

Expressing the posterior distribution $p_{\boldsymbol{\theta}}(y \mid x)$ of EBM by using product rule, applying sum rule and plugging into Equation 2, we have:

$$p_{\boldsymbol{\theta}}(y = i \mid \mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y = i)}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y = i)}{\sum_y p_{\boldsymbol{\theta}}(\mathbf{x}, y = i)} = \frac{\frac{e^{q_i}}{Z_{\boldsymbol{\theta}}}}{\sum_{j=1}^{D_y} \frac{e^{q_j}}{Z_{\boldsymbol{\theta}}}} \tag{11}$$

[5]This is a sample from both $p_{\boldsymbol{\theta}}(\mathbf{x}, y = i)$ and $p_{\boldsymbol{\theta}}(\mathbf{x} \mid y = i)$. $\log p_{\boldsymbol{\theta}}(y)$ is a constant, so $\nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(y) = 0$.

Comparing Equation 10 and Equation 11, we can observe that $\mathbf{z} = \mathbf{q}$ if we force logits obtained by discriminative model not to have $k(\mathbf{x})$ dependent on $\mathbf{x}$ as $Z_{\boldsymbol{\theta}}$ is just a constant which is the same for every $\mathbf{x}$ and y. This was observed by Grathwohl et al. (2019) and the model introduced in Equation 2 is called the Joint Energy-based Model (JEM). They decided to model the joint distribution $\log p_{\boldsymbol{\theta}}(\mathbf{x}, y)$ via the decomposition $\log p_{\boldsymbol{\theta}}(\mathbf{x}) + \log p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$. This factorization is motivated by the fact that the updated (practical) version of SGLD (Equation 5) cannot generate proper samples and this results in a biased gradient estimator of $p_{\boldsymbol{\theta}}(\mathbf{x})$ or $p_{\boldsymbol{\theta}}(\mathbf{x}, y)$. This factorization enables training of an unbiased classifier, because $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$ is not affected by generated samples and is trained in the standard way (Equation 8). Generative part of this model is trained by maximizing $p_{\boldsymbol{\theta}}(\mathbf{x})$ as $\sum_y p_{\boldsymbol{\theta}}(\mathbf{x}, y)$.

The drawback of this approach is that producing samples by SGLD (Equation 5) is time-consuming even if we resort to its updated version. Moreover, speeding up the sampling process by reducing the number of steps t required to produce a sample causes their training to diverge Grathwohl et al. (2019). In fact, the authors declared that the training instability was the most significant flaw of the proposed model. To resolve these instability issues, multiple approaches have been proposed, such as bypassing SGLD during the training Grathwohl et al. (2020); Duvenaud et al. (2021), applying SGLD only to fine-tune samples produced by a different generator Xie et al. (2018), applying SGLD in lower-dimensional hidden space Che et al. (2020) or restricting the model by adding spectral normalization to each layer and regularizing the energy of both generated and real samples Du & Mordatch (2019).

4 STABILIZED JEM

We noticed that the energy of generated images by SGLD is not always comparable to the energy of the training data. Our idea to stabilize the training is guided by the observation that maximization of $\log p(y \mid \mathbf{x})$ using Softmax function (Equation 8) does not diverge during the training while EBM training using Equation 3 or Equation 4 frequently does when improper samples from the distribution are used. Notice that the output of the Softmax is always between 0 and 1 which is not necessarily true for Equation 1 and Equation 2 whose value can be arbitrarily large because its input $\mathbf{x}$ is continuous. This is even more likely to happen when SGLD is not providing competitive samples as their energy becomes very small. We hypothesize that this might be the source of the instability and we propose to find a different way to optimize JEM.

4.1 ADDING MUTUAL INFORMATION TO THE LOSS FUNCTION

We propose to add mutual information (MI) of inputs $\mathbf{X}$ and labels Y distributed according to $p_{\boldsymbol{\theta}}$ defined as

$$\mathrm{I}(\mathbf{X}; \mathrm{Y}) = D_{\mathrm{KL}}(p_{\boldsymbol{\theta}}(\mathbf{x}, y) \| p_{\boldsymbol{\theta}}(\mathbf{x}) p_{\boldsymbol{\theta}}(y)) = \mathbb{E}_{(\mathbf{x},y)\sim p_{\boldsymbol{\theta}}(\mathbf{x},y)} \left[\log \left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y)}{p_{\boldsymbol{\theta}}(\mathbf{x}) p_{\boldsymbol{\theta}}(y)} \right) \right] \tag{12}$$

to the original objective function $\log p_{\boldsymbol{\theta}}(\mathbf{x}, y)$, therefore minimizing the loss function L:

$$-\mathrm{L} = -\mathbb{E}_{(\mathbf{x},y)\sim p_{data}(\mathbf{x},y)} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, y) \right] + \mathbb{E}_{(\mathbf{x},y)\sim p_{\boldsymbol{\theta}}(\mathbf{x},y)} \left[\log \left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y)}{p_{\boldsymbol{\theta}}(\mathbf{x}) p_{\boldsymbol{\theta}}(y)} \right) \right] \tag{13}$$

Maximization of MI can also be interpreted as having a sharp posterior distribution $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$ of each sample $\mathbf{x}$ while maximizing the entropy of marginal distribution $p(y)$ (Equation 14). In other words, the global optimum is reached when each sample belongs only to a single class while all samples together are distributed uniformly with respect to class y.

$$\mathrm{I}(\mathbf{X}; \mathrm{Y}) = \mathbb{E}_{(\mathbf{x},y)\sim p_{\boldsymbol{\theta}}(\mathbf{x},y)} \left[\log \left(\frac{p_{\boldsymbol{\theta}}(y \mid \mathbf{x})}{p_{\boldsymbol{\theta}}(y)} \right) \right] = \mathbb{E}_{(\mathbf{x},y)\sim p_{\boldsymbol{\theta}}(\mathbf{x},y)} \left[\log p_{\boldsymbol{\theta}}(y \mid \mathbf{x}) \right] + \mathrm{H}(p_{\boldsymbol{\theta}}(y)) \tag{14}$$

Approximating the expectation over the model distribution $p_{\boldsymbol{\theta}}$ by the expectation over p_{data} leading to a new loss function $\mathrm{L_a}$ (Equation 15).

$$-\mathrm{L_a} = \mathbb{E}_{p_{data}} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, y) + \log \left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y)}{p_{\boldsymbol{\theta}}(\mathbf{x}) p_{\boldsymbol{\theta}}(y)} \right) \right] = \mathbb{E}_{p_{data}} \left[\log p_{\boldsymbol{\theta}}(y \mid \mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x} \mid y) \right] \tag{15}$$

The rationale behind the approximation of Equation 13 by Equation 15 is that the the optimum of the first RHS term of Equation 13 is reached when $p_{data} = p_{\boldsymbol{\theta}}$ and the term $\mathbb{E}_{p_{data}}\left[\log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x},y)}{p_{\boldsymbol{\theta}}(\mathbf{x})p_{\boldsymbol{\theta}}(y)}\right)\right]$ share the same optimum when the dataset is balanced but does not motivate $p_{\boldsymbol{\theta}}(y)$ to become an uniform distribution. Moreover, even before reaching the optimum, we want samples from p_{data} to be the subset of samples from $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$ and therefore we are performing almost correct updates for train data in order to maximize Equation 13. We show how to get even closer to optimizing the true objective function (Equation 13) in Section A.3.

4.2 MAXIMIZING $p_\theta(\mathbf{x} \mid y)$

Maximizing $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$ can be simply achieved with a softmax function as described in Section 3.2. We are able to evaluate the exact posterior distribution $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$ which is not influenced by Z_θ. As mentioned before, the training is stable and we suggest that it is a consequence of the fact that the train data example is present both in the numerator and denominator and also the fact that we can evaluate the denominator for all values of y. We would like to have the same effect when maximizing $p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$. Expressing $p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$ in Equation 16 to resemble Equation 8.

$$p_{\boldsymbol{\theta}}(\mathbf{x} \mid y) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y)}{\int_{\mathbf{x}} p_{\boldsymbol{\theta}}(\mathbf{x}, y) d\mathbf{x}} = \frac{e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y}}{\int_{\mathbf{x}} e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y} d\mathbf{x}} \tag{16}$$

We could use the gradient[6] of $\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$ to maximize $p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$:

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x} \mid y) = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y - \mathbb{E}_{\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|y)}\left[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y\right] \approx \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y - \frac{1}{N}\sum_i \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x^i})_y \tag{17}$$

Although this resembles Equation 17, using this equation to compute gradients will unfortunately lead to the same instability issues. We suggest an alternative way, by first realizing that in order to calculate the intractable part of Equation 17, we can alternatively use a different distribution $q(\mathbf{x})$ and eventually approximate it using importance sampling. In a special case when the distribution $q(\mathbf{x})$ is (continuous) uniform $q_u(\mathbf{x})$, there is a constant k such that

$$\mathbb{E}_{\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|y)}\left[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y\right] = \mathbb{E}_{\mathbf{x} \sim q(\mathbf{x})}\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)}{q(\mathbf{x})}\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y\right] = \mathrm{k}\ \mathbb{E}_{\mathbf{x} \sim q_u(\mathbf{x})}\left[p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y\right] \tag{18}$$

We hypothesize that in the worst case (complete failure), SGLD draws samples from $q_u(\mathbf{x})$ instead of the desired distribution (in this case $p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$, alternatively $p_{\boldsymbol{\theta}}(\mathbf{x})$ or $p_{\boldsymbol{\theta}}(\mathbf{x}, y)$). If that happens, we should weight samples by $p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$. Approximating expectation of $q_u(\mathbf{x})$ by samples for which we cannot evaluate the exact likelihood of $p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$ in our model, we are forced to use self-normalized variant of importance sampling Bishop & Nasrabadi (2006). Expressing Equation 18 using self-normalized importance sampling, we get

$$\mathrm{k}\,\mathbb{E}_{\mathbf{x} \sim q_u(\mathbf{x})}\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y)}{p_{\boldsymbol{\theta}}(y)}\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y\right] \approx \sum_i^N \frac{\mathrm{k}\,\frac{p_{\boldsymbol{\theta}}(\mathbf{x^i}, y)}{p_{\boldsymbol{\theta}}(y)}}{\mathrm{k}\sum_j^N \frac{p_{\boldsymbol{\theta}}(\mathbf{x^j}, y)}{p_{\boldsymbol{\theta}}(y)}}\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x^i})_y = \sum_i^N \frac{e^{f_{\boldsymbol{\theta}}(\mathbf{x^i})_y}}{\sum_j^N e^{f_{\boldsymbol{\theta}}(\mathbf{x^j})_y}}\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x^i})_y \tag{19}$$

Following this strategy when SGLD fails for **all** samples, it can still be problematic as the denominator of Equation 16 might be much smaller than its numerator. Since the correct denominator of Equation 17 should be much larger than estimated by our (incorrect) samples, we are scaling this gradient compared to the situation when at least one of the samples has comparable energy to train data (this is the consequence of using a self-normalized variant of importance sampling). We can overcome this issue of not estimating the denominator correctly by including the real (training) data point for which we are computing the gradient as one of the negative samples which works as an anchor[7]. The trick of including a positive sample into negative ones **cannot** be used when using an approximation from Equation 17 because we would ignore the gradient of training data by simply subtracting exactly the same gradient. This gives us a stable way to maximize Equation 17 in a case when SGLD produces improper samples. Since the negative energy of generated samples will be

[6] Detailed derivation of Equation 17 can be found in Appendix (Equation 28).

[7] Or as a wall in the analogy introduced in Section 2

much smaller than the real ones, we effectively ignore these samples because the computation of the gradient from Equation 17 becomes almost $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y - \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y$, i.e. zero.

Notice that if EBM tries to reach a likelihood close to 0 for most of the input points $\mathbf{x}$ and we assume that they are effectively exactly 0, we don't need to estimate the gradient based on these. More than that, if we assume that the rest of the input points lie on a low-dimensional manifold either having the same likelihood or the same probability of being sampled by SGLD, the suggested update follows the gradient. We do not think that these assumptions hold and it leads us to an investigation of what we will optimize following suggested updates shall the SGLD provide proper samples.

Before we do that, let us first show what we optimize by using Equation 17 in the case when SGLD generate improper samples distributed according to $q_u(\mathbf{x})$. As seen from Equation 18, we are effectively scaling their gradient proportionally to $\frac{1}{p_{\boldsymbol{\theta}}(\mathbf{x}|y)}$, which will lead to the estimator with undesirably high bias and we suspect it to be the main reason of instability[8].

Notice that when maximizing Equation 8, the value itself cannot be larger than 1 (numerator is not larger than denominator). Practically, this holds because we enumerate over all possibilities of y and the one in the numerator is always part of the denominator. When we try to approximate intractable integral in 16, the same is not true. Therefore, we suggest including the data point from the numerator in the denominator. Moreover, we recommend taking advantage of the fact that other data points in the mini-batch are also samples from p_{data} and the goal of the training is that they are actual samples from $p_{\boldsymbol{\theta}}$. We have observed that including all samples from mini-batch increases the performance[9] as shown in Appendix **??** and is also faster to compute and easier to implement.

Let us now come back and explain the effect of plugging Equation 19 into 17 when SGLD behaves as expected and provides samples from desired distribution (in our case $p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$). In this case, each $\mathbf{x}$ in the denominator of $\frac{e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y}}{\int_{\mathbf{x}} e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y} d\mathbf{x}}$ is multiplied by $p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$ leading to $\frac{e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y}}{\int_{\mathbf{x}} p_{\boldsymbol{\theta}}(\mathbf{x}|y) e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y} d\mathbf{x}}$. Original training of the JEM has the analogy that we push up (trying to increase the likelihood) on training data (positive examples) while we push down (trying to increase the likelihood) on generated (negative) examples[10]. In this analogy, we now push down on the negative examples proportionally to their likelihood more[11]. Better intuition can be gained by realizing that the contribution coming from negative samples (in the limit, when an infinite amount of them are used) will not influence $p_{\boldsymbol{\theta}}(\mathbf{x}^1) \geq p_{\boldsymbol{\theta}}(\mathbf{x}^2)$ for two input data points $\mathbf{x}^1$ and $\mathbf{x}^2$. This means that if the condition holds before the update, it will hold after the update. Our proposed update rule does not have the same property, but in practice, when taking a limited amount of samples, this property is not guaranteed even for the original method. Furthermore, realize that we are updating the parameters $\boldsymbol{\theta}$ in an iterative manner, meaning that compared to the update suggested by the gradient, we push down more on likely $\mathbf{x}$, which results in them becoming less likely. As a consequence, we are going to push proportionally less on them in the next iteration if they are chosen as negative samples. Equation 20 visualizes the resulting change in the update compared to previous technique. Notice that implementation-wise we just first generate SGLD samples, include them in mini-batch and then use the softmax function over the mini-batch.

$$\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y - \frac{1}{N} \sum_i \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x}^{\mathbf{i}})_y \xrightarrow[\text{by}]{\text{replaced}} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y - \sum_i^N \frac{e^{f_{\boldsymbol{\theta}}(\mathbf{x}^{\mathbf{i}})_y}}{\sum_j^N e^{f_{\boldsymbol{\theta}}(\mathbf{x}^{\mathbf{j}})_y}} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x}^{\mathbf{i}})_y \tag{20}$$

The goal of the training is to maximize the expectation over the training data $\mathbb{E}_{p_{data}} \left[\frac{e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y}}{\int_{\mathbf{x}} e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y} d\mathbf{x}} \right]$. The global (over-trained) optimum is reached when the energy of each sample from the dataset is exactly the same (assuming each data point is at maximum once in the dataset) and 0 everywhere else. Assuming that the model $f_{\boldsymbol{\theta}}(\mathbf{x}, y)$ is powerful enough to reach this optimum, this optimum is

[8] One of the recommended tricks to stabilize the training is to ignore samples that have very low likelihood and try to enforce generated samples to have a comparable likelihood to training data e.g. by directly adding the minimization of the difference to the objective function.

[9] We even include samples that belong to a different class according to p_{data} as the desired value under $p_{\boldsymbol{\theta}}$ is anyway 0.

[10] Or vice versa when we talk about energy instead of likelihood.

[11] Notice, that originally we push on the same on all the examples, but these examples are selected based on their likelihood.

the same even when following updates of proposed method in Equation 19.

$$\arg\max_{\boldsymbol{\theta}} \frac{e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y}}{\int_{\mathbf{x}} e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y} d\mathbf{x}} = \arg\max_{\boldsymbol{\theta}} \frac{e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y}}{\int_{\mathbf{x}} p_{\boldsymbol{\theta}}(\mathbf{x} \mid y) e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y} d\mathbf{x}} \tag{21}$$

Unfortunately, when a particular train data point occurs multiple[12] times in the dataset, this will not hold as we will push down on that specific data point more (compared to those less frequent). As this case might not be very important for the real-case scenarios, the real goal of training is not to reach the global over-trained optimum, we are approximating expectations by sampling and these samples will not be exact.

5 Discussion

We can intuitively see this optimization as we are playing a game of showing multiple input samples to our model and asking it to "bet unnormalized log-likelihood", which should be "balanced" among that group of inputs. Afterward, "bets" that were not put on the correct class or were put on negative samples, are lost. The model is trained to regain back an as large portion of "bets" as possible. In this analogy, our approach forces the model to pay more attention to difficult examples. We show later that this method indeed stabilizes the training and, in fact, increases the accuracy of the trained model. We further investigate the reason for the increase in accuracy. To distinguish the model trained by the proposed approach, we refer to it as Stable Training JEM (ST-JEM).

Likewise, an analogy can be made for training. Original training of JEM can be compared to two people (positive and negative samples) lifting a large ball by pushing on it from the opposite sides (the ball can move higher only when both are pushing). When one (negative samples) is not providing comparable force, the force balance is disturbed and the ball falls down (training diverges). We propose that both people should push from the same side using a wall (softmax cannot be higher than 1) on the other side so when the weaker (negative samples) cannot push hard enough, the stronger simply keeps the ball in the same position. It provides time for the weaker to regain its power (details are discussed in Appendix A.4). Moreover, our method could be extended to use any (negative) samples for which low log-likelihood is desired, such as those from out-of-domain datasets or those provided by VERA.

We have performed many experiments to help us better understand the system. Due to the space limitation, we share our insight in the Appendix. As we are not required to use any amount of SGLD samples, we investigate the effect of not having any in Appendix A.1 Experimental part of this work is in Appendix C and Appendix B. The rest of the Appendix describes additional variants and implementation details.

6 Conclusion

We introduced an alternative way of training JEM, called ST-JEM. Unlike the previous training of JEM, our training does not diverge. This is done by making it robust to the effect of improper samples that are the consequence of approximation of the SGLD procedure. Following the proposed training, we obtained systems on two modalities – images and speech. For images, we show that the introduced way of training not only stabilizes the training but also increases the classification accuracy. By investigation, we localized the source of increase of the accuracy, which is not a consequence of the generative part of JEM, but the way of training. Our method allows for any number of SGLD samples per mini-batch and we show that the case of not using any (RES-JEM), leads to the same or even slightly better classification performance than the one reached by ST-JEM. We show that RES-JEM can still be interpreted as JEM that is defined only over the domain of training data points. This assumption makes the model generative only on the theoretical level, as we can neither sample from it, nor evaluate the likelihood outside of the defined domain. In practice, we still use it outside of the defined domain as a classifier that reaches superior accuracy over the discriminative model on CIFAR-10, CIFAR-100, and also when applied to speech. The effectiveness of this approach is most evident for low-resource or more difficult problems. We also

[12] If each data point occurs exactly the same number of times – such as when iterating multiple times over the same dataset – Equation 21 still holds.

demonstrate that, unlike JEM, we are able to increase accuracy by incorporating unlabeled data. Last but not least, we provide a discussion in Appendix C.2 and suggest possible future usage of ST-JEM, which suggests slightly shifting the currently popular approach of training a system on the task and then performing inference by simply forwarding the input through the model with fixed computation time to more human-like. We suggest that many problems can be reformulated as a search problem using trained ST-JEM and we propose to have ST-JEM that will be able to model multiple joint distributions at the same time, which further constrain the search problem and should lead to improved performance by introducing a bias such that the segments that are not speech should not contribute to the process of speaker identification.

REFERENCES

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. *Advances in Neural Information Processing Systems*, 33:12275–12287, 2020.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems*, 32, 2019.

David Duvenaud, Jacob Kelly, Kevin Swersky, Milad Hashemi, Mohammad Norouzi, and Will Grathwohl. No mcmc for me: Amortized samplers for fast and stable training of energy-based models. 2021.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, and Richard Zemel. Learning the stein discrepancy for training and evaluating energy-based models without sampling. In *International Conference on Machine Learning*, pp. 3732–3747. PMLR, 2020.

Ludmila I Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014.

Hao Liu and Pieter Abbeel. Hybrid discriminative-generative training via contrastive learning. *arXiv preprint arXiv:2007.09070*, 2020.

R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6 (3):21–45, 2006. doi: 10.1109/MCAS.2006.1688199.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number CONF. IEEE Signal Processing Society, 2011.

Samik Sadhu and Hynek Hermansky. Continual learning in automatic speech recognition. *Proc. Interspeech 2020*, pp. 1246–1250, 2020.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Martin Sustek, Samik Sadhu, and Hynek Hermansky. Dealing with Unknowns in Continual Learning for End-to-end Automatic Speech Recognition. In *Proc. Interspeech 2022*, pp. 1046–1050, 2022. doi: 10.21437/Interspeech.2022-11139.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Jianwen Xie, Yang Lu, Ruiqi Gao, and Ying Nian Wu. Cooperative learning of energy-based model and latent variable model via mcmc teaching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

A VARIATIONS OF ST-JEM

A.1 RES-JEM

If we don't use any SGLD samples, we end up with a method called RES-JEM. Since we approximate negative samples by mini-batch examples, we can perform the same updates even in the case of including 0 SGLD samples. As this approach seems vague, we explain that it is well-defined and we demonstrate that it actually performs better than a discriminative model maximizing cross-entropy. RES-JEM can be understood as a JEM whose domain is only a subset of $\mathbf{X}$, where training examples lie. Because of that if we use all training examples in a single batch, we directly optimize 13 as our negative samples are now other training examples. The resulting approach is similar to Liu & Abbeel (2020), where they used stored values of other logits from the dataset to re-weight the gradient

A.2 UNLABELED DATA / CLUSTERING / UNSUPERVISED AND SEMI-SUPERVISED LEARNING

We can try to extend Eq. 15 in the case when we are missing labels. A natural extension is to apply an expectation over y for samples for which we do not have ground truth labels.

$$\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\left[\overbrace{\sum_{y} p_{\boldsymbol{\theta}}(y \mid \mathbf{x})}^{\text{replaces } p_{data}(y|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(y \mid \mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)\right]\right] \tag{22}$$

Intuitively, this objective is maximized when the data are grouped into clusters, the number of clusters corresponds to the number of classes, and when sampling from the $p(x)$, the distribution of y should be uniform. Notice that the same cannot be applied to Equation 7 as the naive solution is to assign/collapse everything to a single class. Formed clusters can in general have any meaning and we suggest combining both Equation 15 and Equation 22 in terms of semi-supervised learning to constrain the meaning of the clusters (in speech, clustering could be based on e.g. phonemes, speakers, frequencies, energies or just noise).

A.3 MINIMIZING CROSS-ENTROPY OF CONDITIONALLY GENERATED SAMPLES

By optimizing Eq. 15 instead of Eq. 13, we no longer enforce samples from $p_{\boldsymbol{\theta}}$ to belong to one class. We suggest to replace $p_{\boldsymbol{\theta}}(y)$ by $p_{data}(y)$ which leads to just maximizing posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$ of class y that the generation was conditioned on, because $H(y)$ becomes a constant. Note that it can be simply interpreted as minimizing cross-entropy but also can be seen as encoder-decoder ($y \rightarrow \mathbf{x} \rightarrow y$). Adding the following term to the loss function causes images sampled from $p_{\boldsymbol{\theta}}(x)$ (not conditioned on y) to more likely resemble objects of some particular class [reference to image comparison].

$$-\mathrm{L_s} = \mathbb{E}_{y\sim p_{data}(y)}\mathbb{E}_{\mathbf{x}\sim p_{\boldsymbol{\theta}}(\mathbf{x}|y)}\left[\log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y)}{p_{\boldsymbol{\theta}}(\mathbf{x})p_{\boldsymbol{\theta}}(y)}\right)\right] = \mathbb{E}_{y\sim p_{data}(y)}\mathbb{E}_{(\mathbf{x})\sim p_{\boldsymbol{\theta}}(\mathbf{x}|y)}\left[\log p_{\boldsymbol{\theta}}(y \mid \mathbf{x})\right] \tag{23}$$

Adding the following objective can rarely cause the model to stop producing reasonable samples and we suspect that it happens when SGLD doesn't produce competitive samples yet we still minimize cross-entropy even for them. For that reason, we decided to run the experiments without this loss and only.

A.4 QUALITY OF SGLD SAMPLES AND ADAPTIVE CHANGE OF SGLD HYPER-PARAMETERS DURING TRAINING

"Optimal" hyper-parameters change during the training and we can afford to slightly adapt towards this change because we can afford some exploration (not just exploitation) because if we start pro-

ducing bad samples, it doesn't affect the training... for that period of training we just maximize cross-entropy. In practice, we notice that there might be stages when we are not able to produce reasonable samples and then it gets better again - there is one huge problem - if we focus on getting the best log-likelihood, in the buffer are some samples that are already pretty good and some of them are bad. The hyper-parameters for the bad ones and good ones might be dramatically different in order to increase their negative energy.

B IMAGE EXPERIMENTS

We train our systems ST-JEM and RES-JEM on CIFAR-10 and CIFAR-100 following the same procedure described in JEM. As we did not obtain exactly the same system and we were not able to train the JEM model, we report the results from the paper together with our results. As reported in Table 1, even though our baseline is worse, our ST-JEM and RES-JEM work better than the reported baseline from JEM. We see a small drop in the performance of the ST-JEM compared to RES-JEM. Following the setup from JEM in low-resource settings, we use only 4000 labeled examples for CIFAR-10 and CIFAR-10 and report the largest boost in the accuracy compared to previous systems in Table 2. We also investigate effect of other hyper-parameters in Table 3.

Table 1: Classification accuracies [%] and ECE [%] values for the CIFAR-10 and CIFAR-100 data sets are shown. Apart from the baseline and JEM results reported in Grathwohl et al. (2019), our results with RES-JEM and the DEFAULT ST-JEM models which uses 8 SGLD samples and 20 SGLD steps are also shown.

	Data sets			
	CIFAR-10	**CIFAR-100**	**CIFAR-10**	**CIFAR-100**
Model	**Accuracy %**		**ECE %**	
Baseline (reported)	93.6	74.2	2.7	22.3
Baseline (ours)	93.5	72.0	5.4	22.6
JEM (reported)	92.9	72.2	2.9	4.9
RES-JEM (ours)	94.2	76.7	4.4	7.6
DEFAULT ST-JEM (ours)	93.9	75.7	4.6	11.6

Table 2: The table shows the classification accuracy [%] and ECE [%] values for the CIFAR-10 and CIFAR-100 data sets with 4k labels.

	Datasets (4k labels)			
	CIFAR 10	**CIFAR 100**	**CIFAR 10**	**CIFAR 100**
Model	**Accuracy %**		**ECE %**	
Baseline (reported)	78.0		18.2	
Baseline	77.4	33.9	18.3	51.4
JEM (reported)	74.9		13.7	
RES-JEM	81.1	38.3	13.8	23.0
DEFAULT ST-JEM	79.4	35.4	14.3	19.2
	Including unlabeled data			
RES-JEM	82.53		15.47	

Table 3: Results from variations of the DEFAULT ST-JEM model are shown (see Section **??**). Although for the default ST-JEM configuration, the number of SGLD steps is fixed, further experimentation is done with a variable number of SGLD steps, linearly adding +1 step every 1/10 epochs. Additional experimental results with gradient penalty and incorporation of cross-entropy loss for better image generation are also shown.

ST-JEM variations on CIFAR-100 data set	**Accuracy %**	**ECE %**
DEFAULT - 8 SGLD samples + fixed 20 SGLD steps	75.7	11.6
changing the number of SGLD samples		
2 SGLD samples + gradually increasing SGLD steps from 20 to 40	76.0	10.9
8 SGLD samples + gradually increasing SGLD steps from 20 to 40	75.6	13.3
32 SGLD samples + gradually increasing SGLD steps from 20 to 40	74.3	14.4
changing the range of SGLD steps		
8 SGLD samples + gradually increasing SGLD steps from 20 to 220	76.5	13.5
adding gradient penalty		
8 SGLD samples + gradually increasing SGLD steps from 20 to 220 + gradient penalty	76.1	13.9
adding cross entropy		
DEFAULT + XENT	75.2	13.6

C Speech Experiments

We trained ST-JEM to model the joint distribution of 21 consecutive 80 dimensional Mel-filterbank $\mathbf{x}$ and phoneme y corresponding to the middle frame. We demonstrate the benefits of ST-JEM when combining models trained on different datasets in Section C.1. We further demonstrate that the system is capable of producing reasonable speech by original or modified SGLD procedure. The speech is produced by creating a sequence of 200 frames. For each 21 consecutive frames, we are able to predict joint distribution $p_{\boldsymbol{\theta}}(x, y)$ and we maximize likelihood of the overall sequence by pretending that they are independent as they are already dependent through heavy overlap in $\mathbf{x}$. Conditioned on a sequence of 180 randomly extracted labels from test set. We are, somewhat surprisingly, able to produce understandable speech by first obtaining approximate spectrogram through pseudoinverse of the transformation from spectrogram to mel-filterbank and then reconstructing the phase of that spectrum by using only Grififn-Lim algorithm. We are attaching samples to the submission. When the review process is over, we will publish code together with all samples. By slight modification of SGLD procedure to make it more greedy during the inference, we are able to denoise heavily corrupted speech, samples are also attached to the submission. Preliminary evaluation on few examples showed that while ST-JEM is able to reach about 75% of accuracy, heavily corrupting small portion of input (such that each 21 consecutive frames are affected, results in accuracy around $1 - 20\%$. By performing similar procedure to SGLD, we were typically able to reach accuracy of about $35 - 50\%$. Further and more systematic evaluation is needed to confirm that ST-JEM can work as a denoiser in realistic setting, but our preliminary results suggest so.

C.1 Model Combination

Traditional machines learning systems show unsatisfactory generalization to unknown data domains. The widely prevalent solutions to this unavoidable problem are a variety of data-augmentation methods or, in other cases, just accumulation of training data over as many test conditions as possible for model training. However, an often overlooked solution is that via model combination Polikar (2006); Kuncheva (2014); Sadhu & Hermansky (2020); Sustek et al. (2022).

Given two classifiers trained on different data domains, under an unknown test condition the aim is to obtain a *weighted combination* of the posterior distribution from the two models that aids in better classification accuracy. That said, the main challenge in this approach lies in finding the best combination strategy. In the following section we describe our combination strategy with ST-JEM based speech recognition systems.

C.1.1 COMBINATION OF AUTOMATIC SPEECH RECOGNITION (ASR) SYSTEMS

Hybrid automatic speech recognition systems require conditional likelihood values $p(\mathbf{x}|y)$ for every feature vector $\mathbf{x}$ computed at a desired temporal sampling rate over a considerable duration of speech for a large ($\approx$ 3000) number of tri-phonetic states y of a Hidden Markov Model (HMM). During inference, these likelihood values are then passed onto a decoding graph to find the best path evaluated by likelihoods and constrained by context, lexicon and grammar to obtain text Povey et al. (2011).

Consider two ST-JEMs trained on two different data sets indexed by 1 and 2. For our experiments the two data sets used are Wall Street Journal (WSJ) and REVERB which comprises of clean read speech and simulated reverberated speech respectively. The input features $\mathbf{x} \in \mathbb{R}^{80\times 9}$ are obtained by concatenating 80 dimensional mel-filterbank features over 9 contiguous frames sampled at 100 Hz in time with 3376 triphoneme states i.e., $y \in \{0, 1, 2, \dots 3375\}$. Assigning $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$ to be the learnt parameters from the first and second data set respectively and $\boldsymbol{\theta_{comb}}$ to be the combined parameter set, we propose the following combination strategy.

$$\begin{aligned} p_{\boldsymbol{\theta_{comb}}}(\mathbf{x}, y) &= \frac{p_{\boldsymbol{\theta_1}}(\mathbf{x}, y) + p_{\boldsymbol{\theta_2}}(\mathbf{x}, y)}{2} \\ &= \frac{e^{f_{\boldsymbol{\theta_1}}(\mathbf{x})_y}}{2Z(\boldsymbol{\theta_1})} + \frac{e^{f_{\boldsymbol{\theta_2}}(\mathbf{x})_y}}{2Z(\boldsymbol{\theta_2})} \end{aligned} \tag{24}$$

To understand the rational behind combining joint distributions, observe that $p_{\boldsymbol{\theta_{comb}}}(\mathbf{x}, y) = \frac{p_{\boldsymbol{\theta_1}}(\mathbf{x})p_{\boldsymbol{\theta_1}}(y|\mathbf{x}) + p_{\boldsymbol{\theta_2}}(\mathbf{x})p_{\boldsymbol{\theta_2}}(y|\mathbf{x})}{2}$. Therefore $p_{\boldsymbol{\theta_{comb}}}(\mathbf{x}, y)$ automatically combines the posterior distributions from each model *weighted* by the likelihood of a given feature vector $\mathbf{x}$ from each model. For an unknown test feature vector $\mathbf{x}_{test}$, the relative value of $p_{\boldsymbol{\theta_1}}(\mathbf{x}_{test})$ vs $p_{\boldsymbol{\theta_2}}(\mathbf{x}_{test})$ represents how well the two ST-JEMs recognize $\mathbf{x}_{test}$ to match with their individual training conditions - a higher likelihood indicating a better match.

In our experiments we observed that the partition functions from two different JEM-STs have a very similar range of values obtained over several SGLD samples which leads us to safely assume $Z(\boldsymbol{\theta_1}) \approx Z(\boldsymbol{\theta_2})$. This simplification is further motivated by the fact that preserving Equation 24 as is for the combination strategy makes no consequential change in the final ASR performance.

For some constant value C, assuming $Z(\boldsymbol{\theta_1}) = Z(\boldsymbol{\theta_2}) = C$, we get

$$\log p_{\boldsymbol{\theta_{comb}}}(\mathbf{x}, y) = \log(e^{f_{\boldsymbol{\theta_1}}(\mathbf{x})_y} + e^{f_{\boldsymbol{\theta_2}}(\mathbf{x})_y}) - \log 2C \tag{25}$$

The constant C being simply a scaling factor, and given the prior probability distribution $p(y)$, the conditional log likelihoods required by the decoding graph can be obtained as follows

$$\log p_{\boldsymbol{\theta_{comb}}}(\mathbf{x}|y) \equiv \log(e^{f_{\boldsymbol{\theta_1}}(\mathbf{x})_y} + e^{f_{\boldsymbol{\theta_2}}(\mathbf{x})_y}) - \log p(y). \tag{26}$$

Note that our combination strategy comes down to computing the logssumexp of the logits across models and can be easily generalized to more than two ST-JEMs as in Equation 26.

$$\log p_{\boldsymbol{\theta_{comb}}}(\mathbf{x}|y) \equiv \log \sum_i \exp e^{f_{\boldsymbol{\theta_i}}(\mathbf{x})_y} - \log p(y) \tag{27}$$

Table 4 shows a comparison of Word Error Rates (WER %) of baseline and ST-JEM model combination. The baseline combination follows the same principle as Equation 26 and 27 where the

ST-JEM logits are replaced by the logits from a standard classifier with the same architecture as the ST-JEM model.

Table 4: ASR Word Error Rate [%] results are shown for individual ST-JEM models trained on WSJ and REVERB data sets together with the result of model combination. The ST-JEM combination results when compared with the baseline combination performance shows the advantage of ST-JEM model combination for robust ASR.

Test set	Model	WER %		
		Testing Model		
		WSJ	REVERB	Combination
WSJ	Baseline	9.0	29.7	10.1
	ST-JEM	8.8	27.5	9.8
REVERB	Baseline	32.0	8.1	8.3
	ST-JEM	31.1	7.5	7.4

Except for slight improvement in WER, we can observe very well calibrated system as shown in Figure 1.

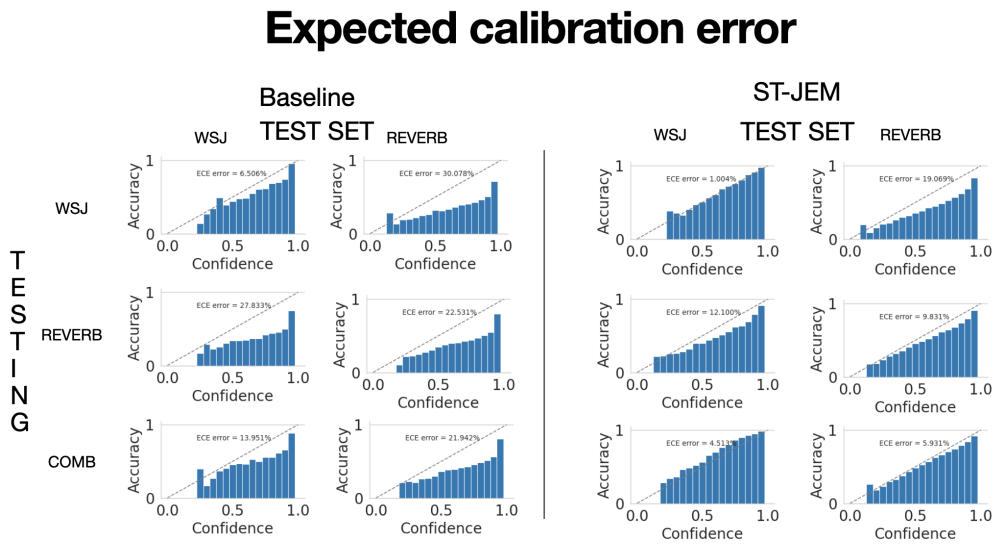

Figure 1: Comparison of ECE between Baseline system and proposed ST-JEM evaluated on WSJ and REVERB datasets. The last row corresponds to described combination of the systems.

.

C.2 Speech JEM as multi-purpose system

The general view of machine learning models is to train them on the same task that they are going to perform. We want to discuss the potential future use of JEM trained in a stable way. We suggest that JEM can model multiple joint distributions at the same time, e.g. joint distribution of inputs $\mathbf{x}$ and phonemes ph and at the same time joint distribution of $\mathbf{x}$ and the identity of the spk. We noticed that, when we try to generate speech from the model, it has no notion of speaker identity as it was never exposed to a sequence longer than 21 frames. Having the access to speaker identity, we can condition our generation on the fact that the likelihood of the speaker for each frame needs to be high and also the same. If the model is powerful enough to learn this complex distribution, it might function as text-to-speech (TTS). This would further allow us to perform voice conversion. Another interesting domain for JEM could be source separation, as we could follow some sort of

updated version of SGLD to iteratively separate speech into two parts that sum into the original speech, where in each iteration, separated speech would be conditioned to have the high likelihood for that particular speaker. Next, the straightforward application is inpainting, we noticed that JEM is capable of very interesting results when exposed to an image of e.g. airplane and we perform SGLD conditioned on the cat class as the model is able to change the class of the image while visually on the pixel level, the resulting image is very close to the original one. This might open the possibility of more sophisticated changes than just speech conversion.

We found interesting parallel between the inference through SGLD in JEM and human reasoning. Using more sophisticated inference with adaptive computation time compared to just forwarding the input through a model is more aligned with how people think. Moreover, generative part of JEM can work as a proxy to the ability of people being self-aware of when they do not know. This motivates future work on these models even more.

D DERIVATIONS

Expressing gradient of $\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid y)$ by substituting term from Equation 29:

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x} \mid y) = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y - \nabla_{\boldsymbol{\theta}} \log \int_{\mathbf{x}} e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y} d\mathbf{x} = \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y - \mathbb{E}_{\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|y)} \left[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y \right] \tag{28}$$

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \log \int_{\mathbf{x}} e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y d\mathbf{x}} &= \frac{\int_{\mathbf{x}} e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}(\mathbf{x})_y} d\mathbf{x}}{\int_{\mathbf{x}} e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y} d\mathbf{x}} = \int_{\mathbf{x}} \frac{e^{f_{\boldsymbol{\theta}}(\mathbf{x})_y}}{Z_{\boldsymbol{\theta}}} \frac{\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}(\mathbf{x})_y}}{p_{\boldsymbol{\theta}}(y)} d\mathbf{x} \\ &= \int_{\mathbf{x}} \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, y)}{p_{\boldsymbol{\theta}}(y)} \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}(\mathbf{x})_y} = \mathbb{E}_{\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|y)} \left[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\mathbf{x})_y \right] \end{aligned} \tag{29}$$