Pixel Recurrent Neural Networks

1 Citation

Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." arXiv preprint arXiv:1601.06759 (2016).

`https://arxiv.org/pdf/1601.06759v2.pdf`

2 Abstract

We generate images with a recurrent neural net (RNN) that predicts one pixel at a time.

3 Introduction

Variational autoencoders (VAEs) are popular generative models for images, but they can only generate examples rather than compute the probability density. Our PixelRNN uses 12 LSTMs to predict one pixel at a time. We develop two layer types - Row LSTMs and Diagonal BiLSTMs. We also make a PixelCNN, which uses masked convolutions to indicate which other pixels the model can use as dependencies - this is much faster than the LSTM models. We use residual connections in our networks and model pixels as discrete variables with a multinomial distribution (a softmax comptutes the parameters of the multinomial).

4 Model

Given $n \times n$ image $\boldsymbol{x}$, we seek to estimate and sample from $p(\boldsymbol{x})$. We can model this with:

$$p(\boldsymbol{x}) = \prod_{i=1}^{n^2} p(x_{i,R}|\boldsymbol{x}_{<i})p(x_{i,G}|\boldsymbol{x}_{<i}, x_{i,R})p(x_{i,B}|\boldsymbol{x}_{<i}, x_{i,R}, x_{i,G}) \tag{1}$$

Our pixels are modeled as a 256 category multinomial distribution whose parameters come from a 256-way softmax.

5 Pixel Recurrent Neural Networks

A Row LSTM moves left-to-right top-to-bottom generating pixels based on the previous pixels. We can make it consider the k pixels in the previous row as input with a 1D convolution. To prevent the model from using future pixels as dependencies to the current pixel, we can use a mask (i.e. set weight to zero

when we are not allowed to use a particular input). The Row LSTM is fast, but its field of view does not consider all previous pixels and it is unidirectional rather than bidirectional.

The Diagonal BiLSTM fixes the above issues by scanning from one corner to the other. To enable this, we need to warp (skew) the pixels so that the convolution is easy to apply. It crosses the diagonal from both directions (bidiretional).

We use residual connections to improve convergence speed of training.

As described before, we need to use masked convolution to restrict the input used by a computation. For example, based on the probability density formulation above, we cannot predict the green channel of a pixel using its blue channel as an input.

Our Row LSTM and Diagonal BiLSTM must be computed sequentially and their receptive field is unbounded (because the LSTM hidden state is a function of all previous inputs). The PixelCNN has a fixed-size, but large receptive field and uses a bunch of masked convolutions (no LSTMs). This makes it much faster to compute.

We also create a Multi-scale PixelRNN that has PixelRNNs at different scales. The smallest scale PixelRNN generates an image. This gets upsampled with deconvolutional layers and is fed as input to the higher scale PixelRNN.

6 Specifications of Models

We have two kinds of masks (the only difference is whether the input can have a self-weight). We use ReLU. We use more hidden units for CIFAR-10 and ImageNet than for MNIST.

7 Experiments

We train on log-likelihood. We need to do a little work because our distribution is discrete. We train with RMSProp on a GPU with the Torch framework. For MNIST and CIFAR-10, we use minibatches of size 16. For ImageNet, we use as large a batch as possible. The initial recurrent state is a learned bias.

Using our softmax multinomial distribution works much better than using a continuous mixture of Gaussians. Residual connections also help the model.

The Diagonal BiLSTM beats the Row LSTM, PixelCNN, and previous state of the art approaches.

Qualitatively, our generated ImageNet pictures capture some aspects of natural images, but are not realistic. This is probably because the ImageNet dataset is very diverse, making it difficult to learn.

8 Conclusion

Our PixelRNN and PixelCNN models generate one pixel at a time. We use masked convolutions to control dependencies and we use a softmax to parameterize a multinomial distribution for the pixels. Using larger models and larger datasets will probably improve results a lot.