論文紹介

Pixel Recurrent Neural Networks

ICML2016 読み会 @ ドワンゴ

得居 誠也, Preferred Networks & 東京大学

概要

Pixel Recurrent Neural Networks.

Aäron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu
(all from Google DeepMind)

ICML2016 のベストペーパー 3 本のうちのひとつ

-要約-
ピクセル・チャンネル単位の自己回帰型モデル（RNN 言語モデルに似た確率分布）であって並列化によって高速に勾配計算ができるものを提案し、MNIST, CIFAR-10, ImageNet (32x32, 64x64) で非常に高い対数尤度を達成した

画像の生成モデル

- 訓練データ：大量の画像（グレースケールや RGB 画像）
  - 各画像は Channel x Height x Width の多次元配列で表される
- タスク：訓練データを生成する分布を推定する
  - 確率分布のモデルを考えて、訓練データの生成分布に一番近くなるようなパラメータを推定する
  - **対数尤度最大化**の問題として解くのが一般的

対数尤度

$$\max_{\theta} \frac{1}{N} \sum_{j=1}^{N} \log p_\theta(x^{(j)})$$

パラメータ　生成モデル　訓練データ

- 主な目的: **(1) 新しいデータの生成** (2) 潜在変数の推定

今日はこちら

画像の生成モデル：３つのアプローチ
（昨今流行っているもの、本当はほかにもある）

変分推論（変分 AutoEncoder など）
- 潜在変数 z をつかって $p(z), p(x|z)$ をモデル化
- x だけから学習するのに $p(z|x)$ が必要だが一般に計算困難
- そこで別のモデル $q(z|x)$ をつかってこれを近似する

敵対的生成モデル（Generative Adversarial Nets）
- 潜在変数 z をつかって $x = G(z)$ とモデル化
- こうしてできた $p(x)$ 全体と真の分布を分別する識別器 $D(x)$ を学習しつつ、識別できなくするように G も学習

自己回帰モデル（今日の話）
- 分布を次のように分解： $p(x) = \prod_{i=1}^{d} p(x_i|x_1, \ldots, x_{i-1})$
- 各因子を NN などでモデル化

画像の生成モデル：３つのアプローチ 長所・短所の比較

	変分推論	敵対的生成モデル	自己回帰モデル
訓練方法	**変分下界の最大化**	min max 交互最適化	**対数尤度の最大化**
潜在変数の推論	**近似分布が得られる**	あとから学習する必要	N/A
サンプリング	**低コスト**	**低コスト**	高コスト
評価方法	**変分下界**	定量的評価が困難	**対数尤度**
モデルの設計	生成モデル 推論モデル	生成関数 識別モデル	条件付き分布たくさん （学習の高速化に要工夫：**本論文の貢献**）

自己回帰モデル

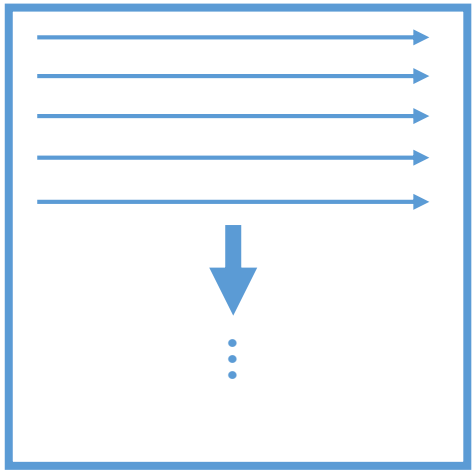
☺ **対数尤度が直接最大化でき、評価もできるところが良い**

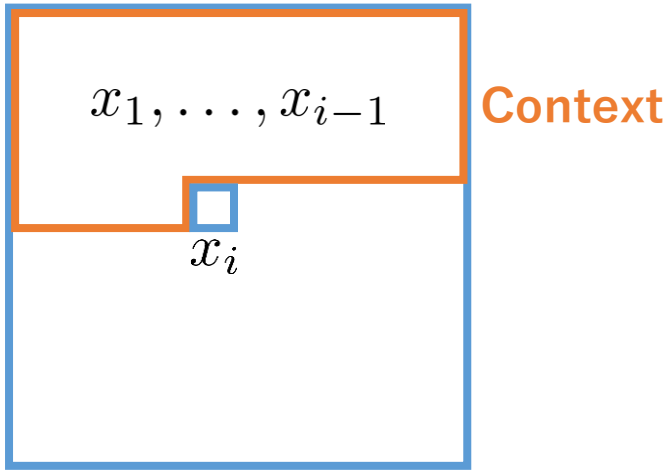
☹ サンプリングが高コストなのが短所

☹ ナイーブなモデル化では学習も高コスト
→ masking (MADE [Germain+, ICML'15]) によって並列化可能に

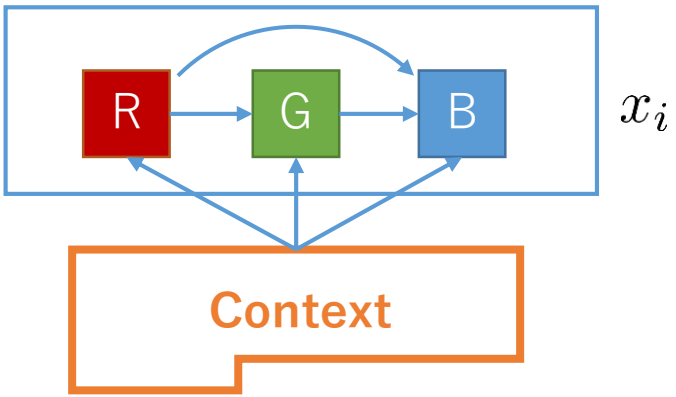
画像の自己回帰モデル

ピクセルを上から下に、行ごとに左から右に生成

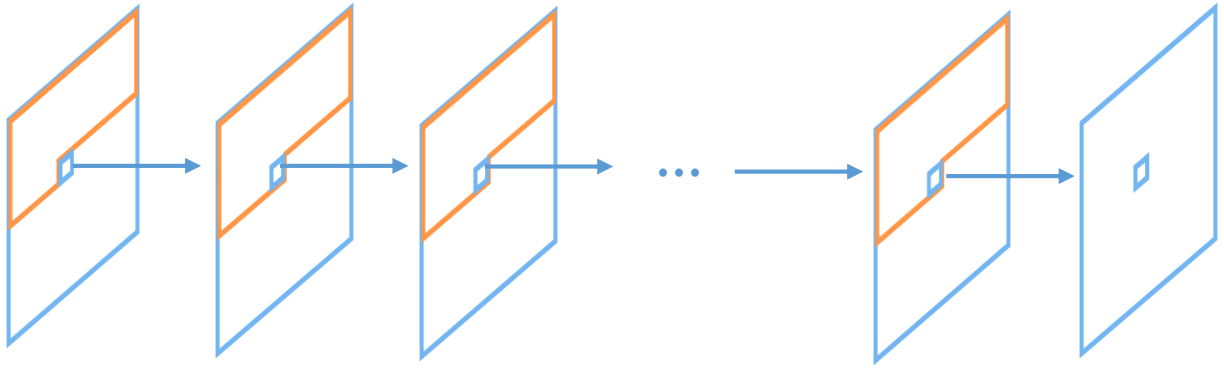




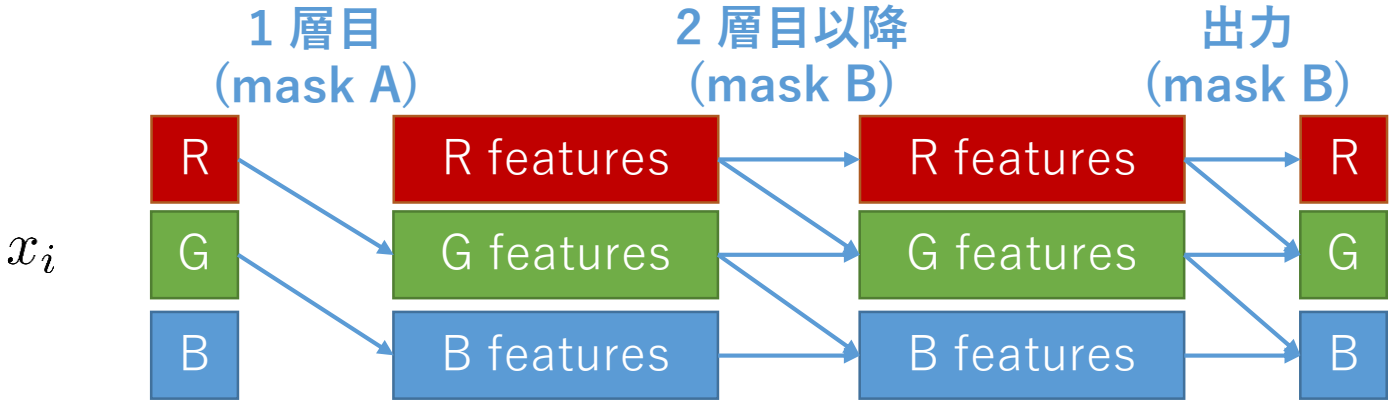
RGB は R → G → B の順に生成



提案モデルの基本：制限された結合



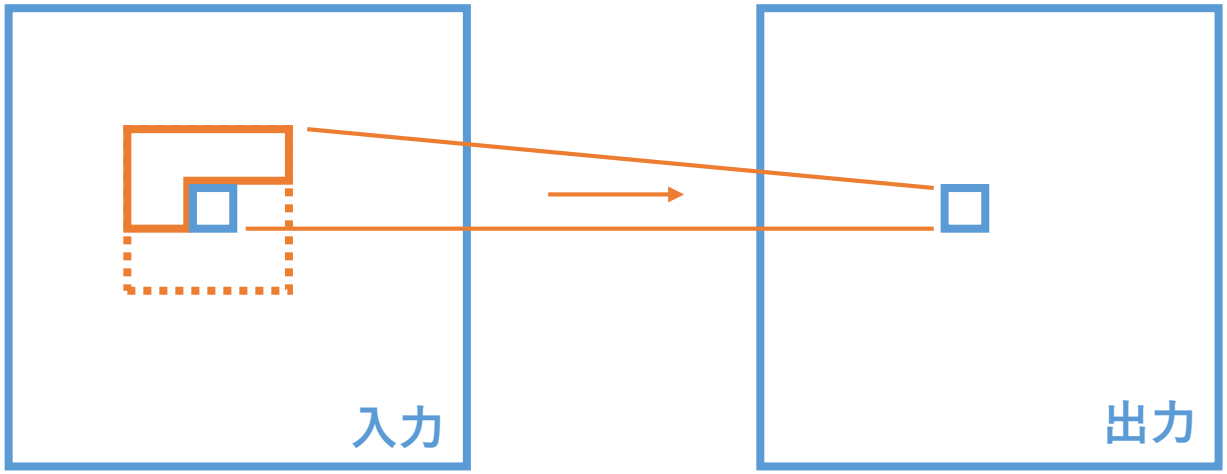
- 同じ spatial size のまま深くしていく (チャンネルは RGB のグループに分割) が、context 外からの結合はなくす (masking)
- 2 層目以降では**同じ位置の同じチャンネルの入力は見ても良い**



cf. MADE [Germain+, ICML'15]

提案モデル１：PixelCNN

各レイヤーは、カーネルの上半分だけ使う 2D convolution

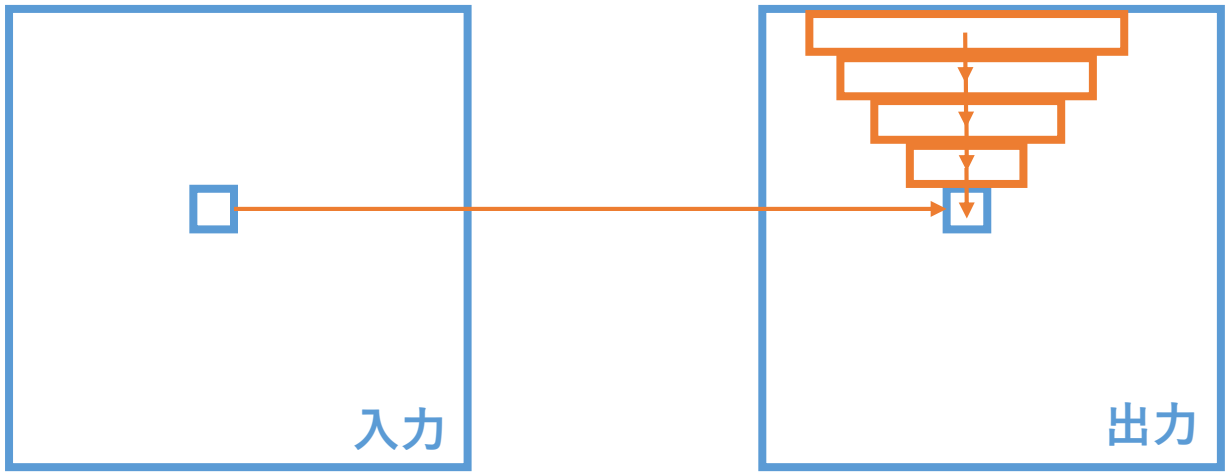


普通の 2D convolution と同じく、位置ごとにパラメータは共通

- 尤度計算は高速（普通の ConvNet と同じくらい）
- Context をフルに使えない、スケール依存

提案モデル２：PixelRNN (Row LSTM)

各行ごとに上から下へ 1D Convolution + LSTM

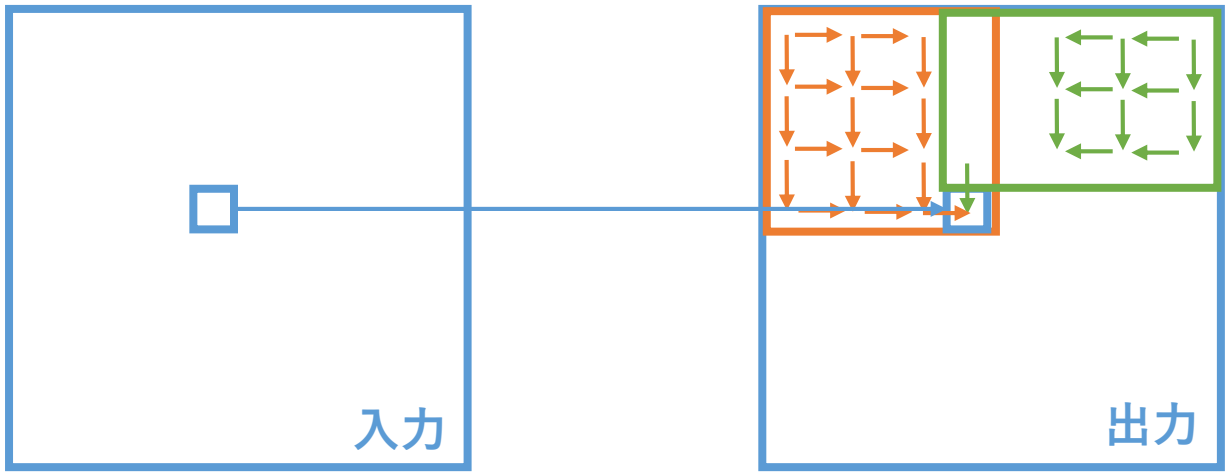


1D Convolution+LSTM なので、位置ごとにパラメータは共通

- 上方向の Context をフルに使える
- 利用可能な Context をまだすべて使えない

提案モデル３：PixelRNN (Diagonal BiLSTM)

2D LSTM を 左→右 と 右→左 の両方向に流す

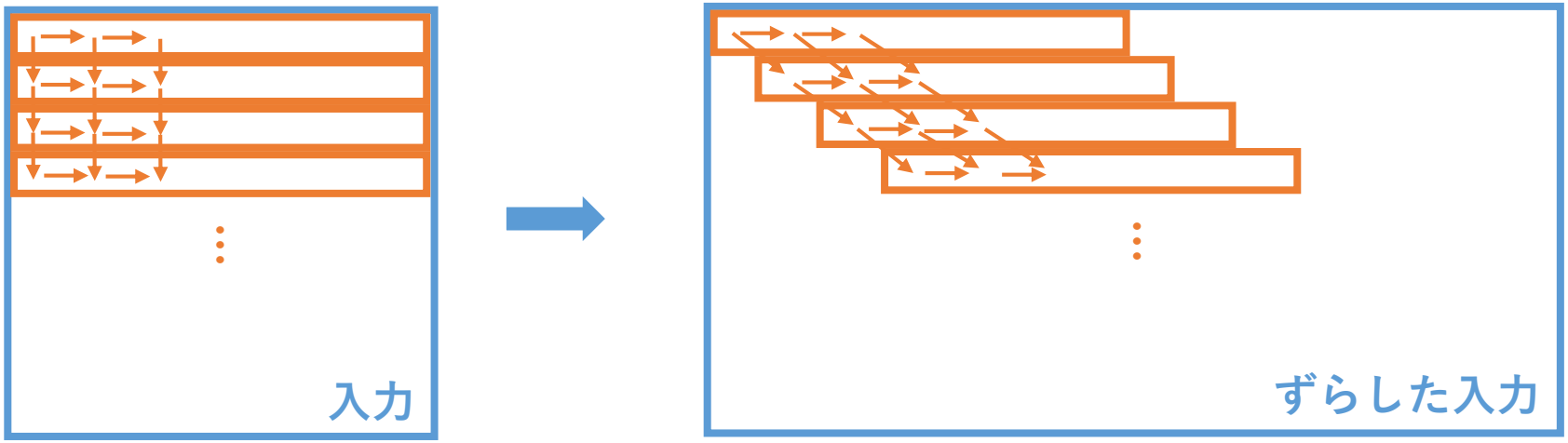


1D convolution を使った高速化テクニックがある (次スライド)

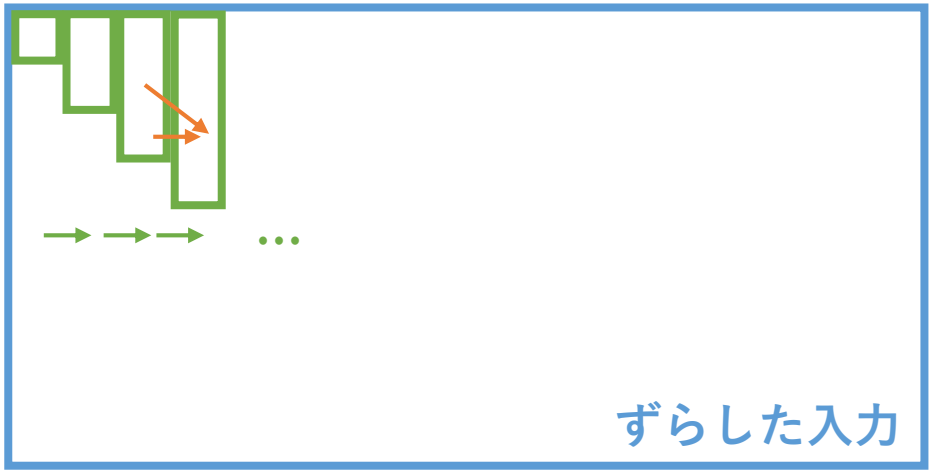
- Context をフルに扱える
- 計算コストは少し高い

Diagonal BiLSTM の実装

各行を 1 pixel ずつずらす



カーネルサイズ 2 の
1D convolutional LSTM を
左から右にかけていくのと等価



出力：ピクセルRGBの 256 値分類

最終層では、実数でピクセル値を予測するのではなく、RGB の各値を 0 – 255 の 256 値の中から一つ予測する（多クラス分類）

よって最終層は Softmax。

2 つの問題を解決：

- 画像の分布をガウシアン等でモデル化する場合の問題点だった「ボケた画像を生成しやすい」問題の回避
- ガウシアン等ではモデル化できない多峰性分布を表せる

ピクセル値の大小に関する情報は一切いれていない
→ データが多ければ問題ない

256-way softmax の出力例

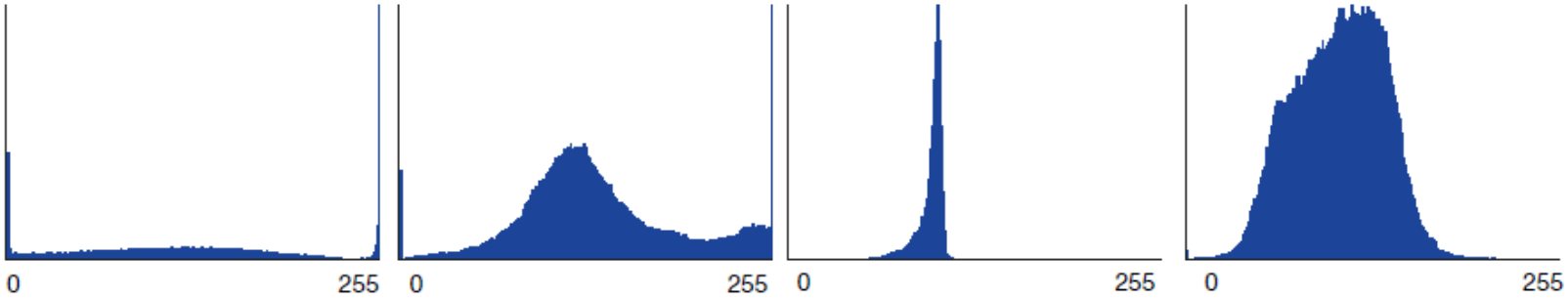


Figure 6. Example softmax activations from the model. The left-most shows the distribution of the first pixel red value (first value to sample).

- 大小関係の情報を入れなくてもなめらかな分布が得られる
- 0 や 255 が出やすい、というような多峰性を獲得
- 区間 [0, 255] の外の値を決して出力しないという利点もある

Residual Block を使った階層化

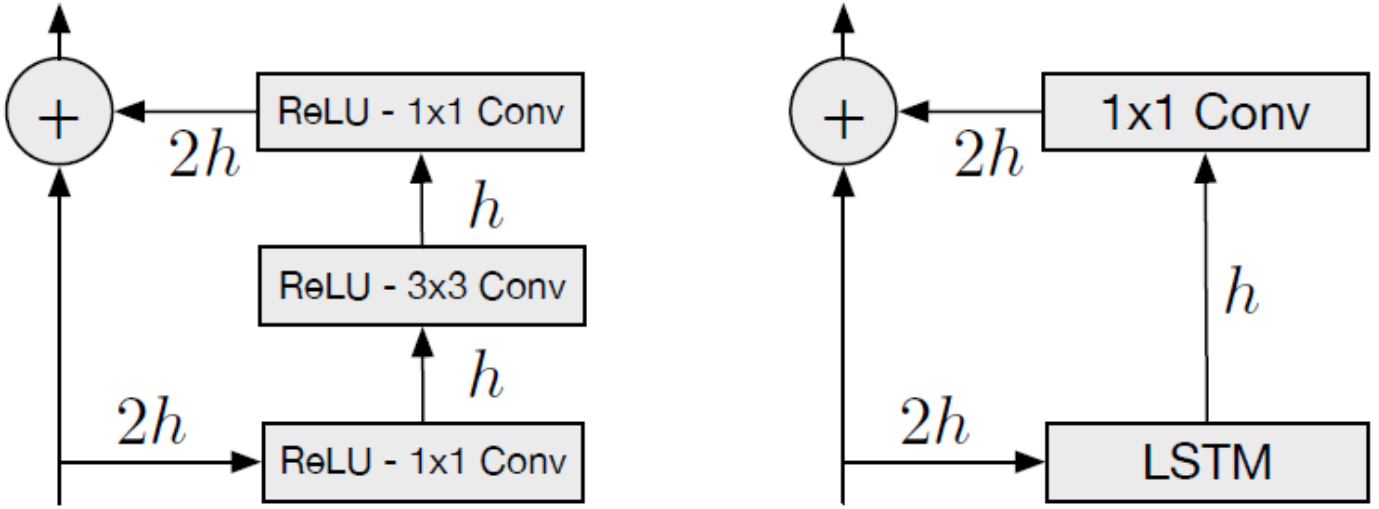


Figure 5. Residual blocks for a PixelCNN (left) and PixelRNNs.

- PixelCNN や PixelRNN を深くするときに Residual Block を使う
- PixelRNN の場合、後ろにだけ projection layer を入れている

最終的なアーキテクチャ

PixelCNN	Row LSTM	Diagonal BiLSTM
7×7 conv mask A		
Multiple residual blocks: (see fig 5)		
Conv 3×3 mask B	Row LSTM i-s: 3×1 mask B s-s: 3×1 no mask	Diagonal BiLSTM i-s: 1×1 mask B s-s: 1×2 no mask
ReLU followed by 1×1 conv, mask B (2 layers)		
256-way Softmax for each RGB color (Natural images) or Sigmoid (MNIST)		

Table 1. Details of the architectures. In the LSTM architectures i-s and s-s stand for input-state and state-state convolutions.

MNIST: 7 レイヤー (Diag. BiLSTM)

CIFAR-10: PixelCNN=15 レイヤー, PixelRNN=12 レイヤー

ImageNet (32x32): 12 レイヤー (Row LSTM)

ImageNet (64x64): 4 レイヤー (Row LSTM, no residual)

実験

Binarized MNIST
- 手書き文字データセット
- 過去手法との比較のために実施。二値画像。

CIFAR-10
- 32x32 のカラー画像データセット（訓練データ 50,000 枚）

ILSVRC (ImageNet)
- 大規模画像認識コンペのデータセット、約 128 万枚
- 今回は 32x32, 64x64 に縮小したものを使用

実験結果：負対数尤度の比較

Model	NLL Test
DBM 2hl [1]:	≈ 84.62
DBN 2hl [2]:	≈ 84.55
NADE [3]:	88.33
EoNADE 2hl (128 orderings) [3]:	85.10
EoNADE-5 2hl (128 orderings) [4]:	84.68
DLGM [5]:	≈ 86.60
DLGM 8 leapfrog steps [6]:	≈ 85.51
DARN 1hl [7]:	≈ 84.13
MADE 2hl (32 masks) [8]:	86.64
DRAW [9]:	≤ 80.97
PixelCNN:	81.30
Row LSTM:	80.54
Diagonal BiLSTM (1 layer, $h = 32$):	**80.75**
Diagonal BiLSTM (7 layers, $h = 16$):	**79.20**

Table 4. Test set performance of different models on MNIST in *nats* (negative log-likelihood). Prior results taken from [1]

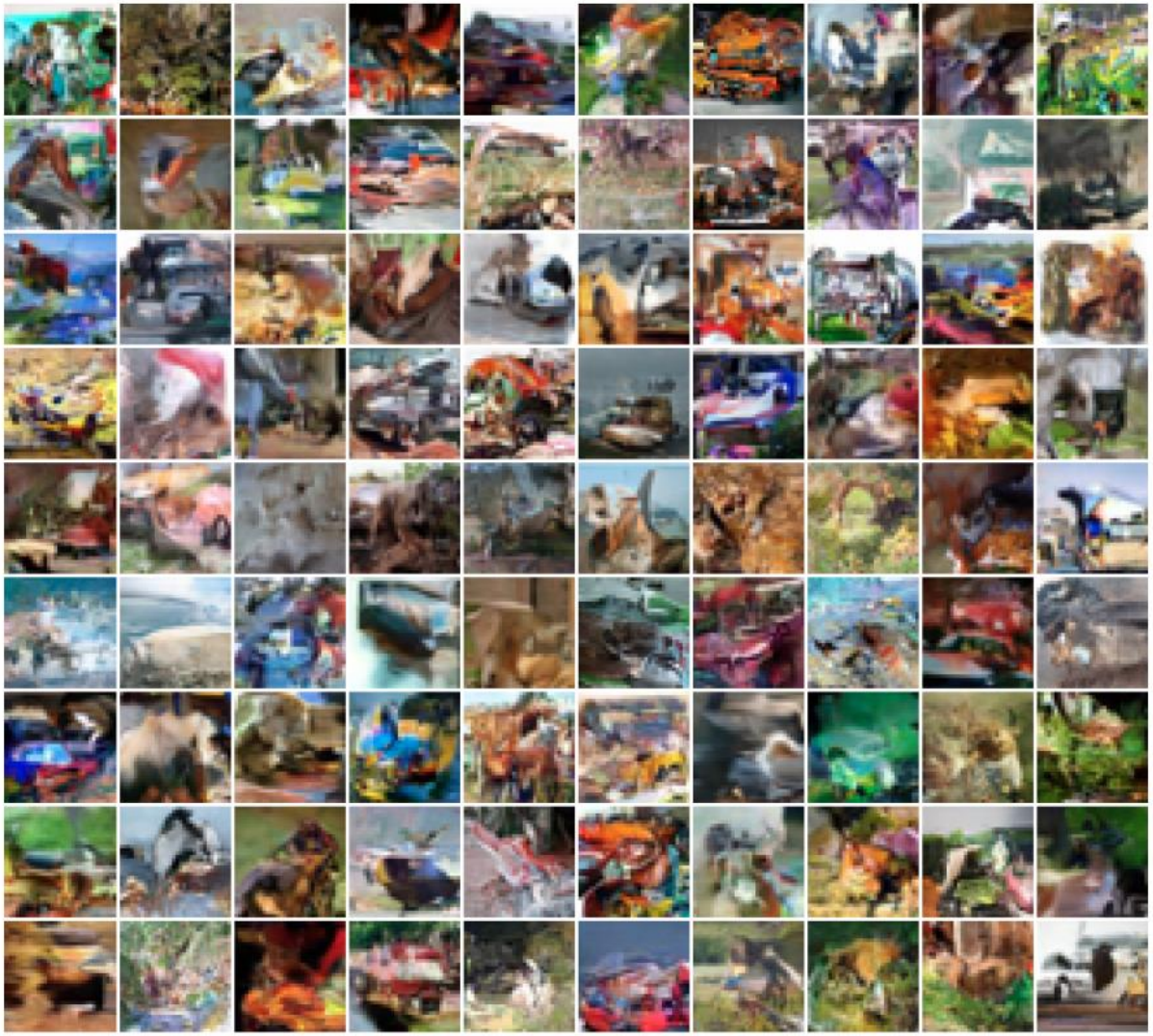
Model	NLL Test (Train)
Uniform Distribution:	8.00
Multivariate Gaussian:	4.70
NICE [1]:	4.48
Deep Diffusion [2]:	4.20
Deep GMMs [3]:	4.00
RIDE [4]:	3.47
PixelCNN:	3.14 (3.08)
Row LSTM:	3.07 (3.00)
Diagonal BiLSTM:	**3.00** (2.93)

Table 5. Test set performance of different models on CIFAR-10 in *bits/dim*. For our models we give training performance in brackets. [1] (Dinh et al., 2014), [2] (Sohl-Dickstein et al., 2015), [3] (van den Oord & Schrauwen, 2014a), [4] personal communication (Theis & Bethge, 2015).

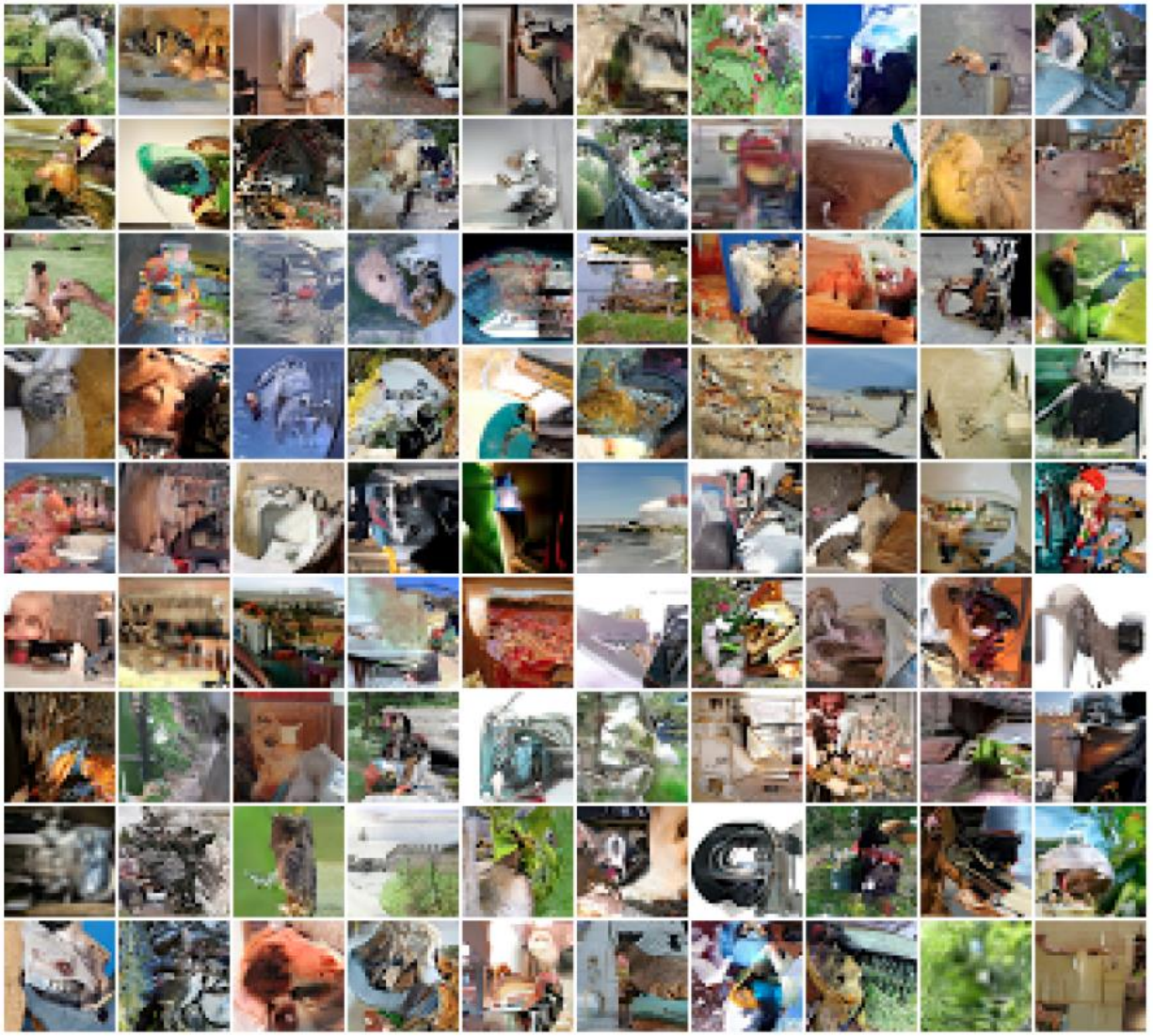
Image size	NLL Validation (Train)
32x32:	3.86 (3.83)
64x64:	3.63 (3.57)

Table 6. Negative log-likelihood performance on 32×32 and 64×64 ImageNet in *bits/dim*.

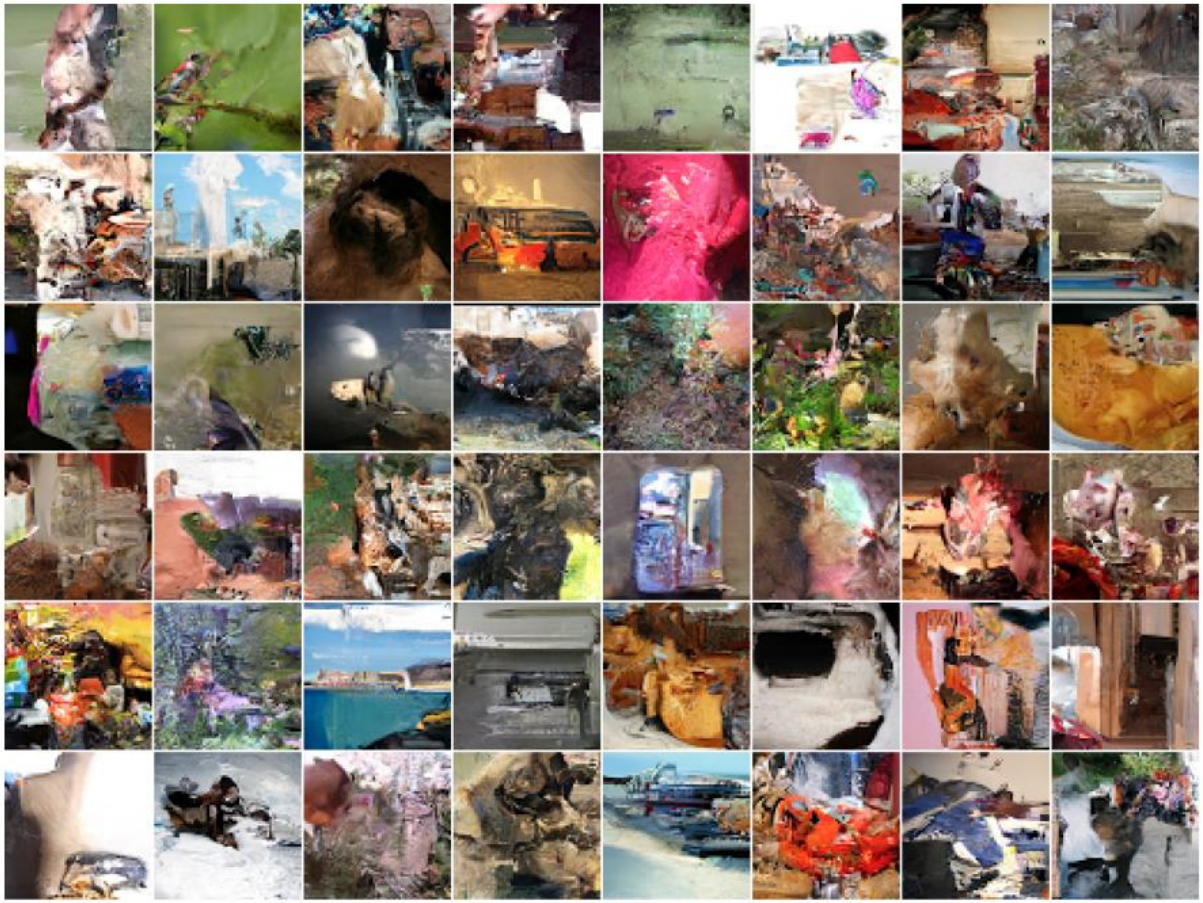
実験結果：CIFAR-10 生成例



実験結果：ImageNet 32x32 生成例



実験結果：ImageNet 64x64 (multi-scale)



半分のサイズの画像を生成してから、それで条件付けて大きな画像を生成する → 対数尤度はあまり変わらないが大域的により整合性のある画像が生成できるようになった

まとめ

- GAN に対する DCGAN のような感じで、自己回帰モデルにおける画像生成のモデル化を提案した
- 学習時には各条件付き分布を並列に計算することができる
- MNIST, CIFAR-10 で SOTA な対数尤度を達成
- 今までやられていなかった ImageNet での尤度評価も実施
- DCGAN などと competitive な画像生成ができている

- GAN と違い定量的な評価が可能なのが強み
- VAE と違い対数尤度を正確に計算できるのが強み
- これらと違い 1 pixel ずつ生成しないといけない（生成時は計算が並列化しづらく遅い、はず）のが弱点