Глубинное обучение
Лекция 8: Вероятностные модели и нейросети

Лектор: Антон Осокин

ФКН ВШЭ, 2018

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

План лекции

- Нейросети как вероятностные распределения
- Авторегрессионные модели
  – Seq2seq, ByteNet, PixelCNN
- Неявные вероятностные модели
  – GAN
- Вероятностные модели со скрытыми переменными
  – VAE

Softmax выдает распределение!

- Обычная функция потерь для классификации – кросс-энтропия

$$\ell(f(x,\theta),y) = -\log\left(\frac{\exp f_y(x,\theta)}{\sum_{s=1}^K \exp f_s(x,\theta)}\right)$$

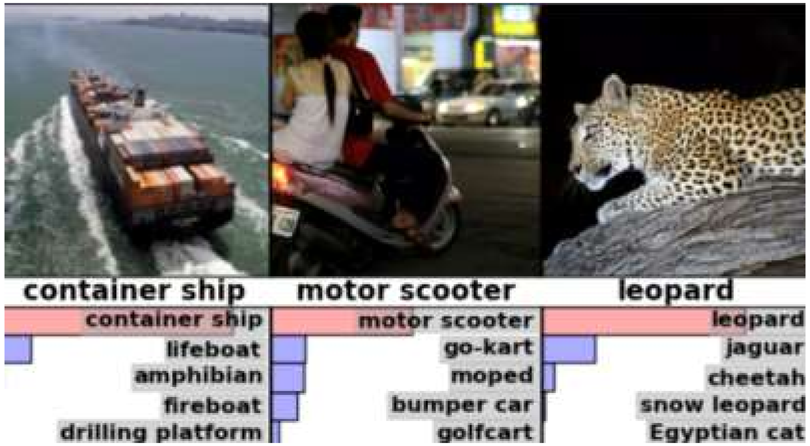
- Softmax можно интерпретировать как вероятности

$$p(y|x,\theta) = \frac{\exp f_y(x,\theta)}{\sum_{s=1}^K \exp f_s(x,\theta)}$$

- Правдоподобие – вероятность правильных ответов

- Вероятности показывают «уверенность» сети

- Настоящие вероятности?



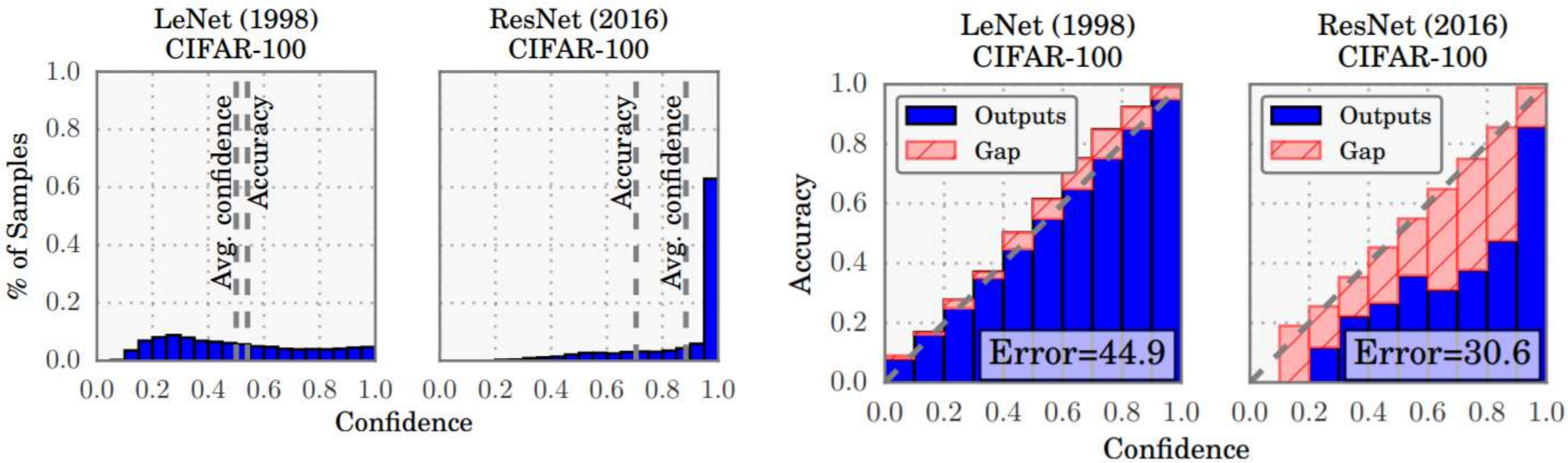
Калиброванность вероятностей

[Guo et al., 2017]

- Калиброванность – удобное свойство вероятностей
- Калиброванность означает, что если вероятность *p*, то ответ правильный – в доли случаев *p*

$$P\left(y = y_{\text{true}} \middle| p(y|x,\theta) = p\right) = p$$

- Выдают ли нейросети калиброванные вероятности?

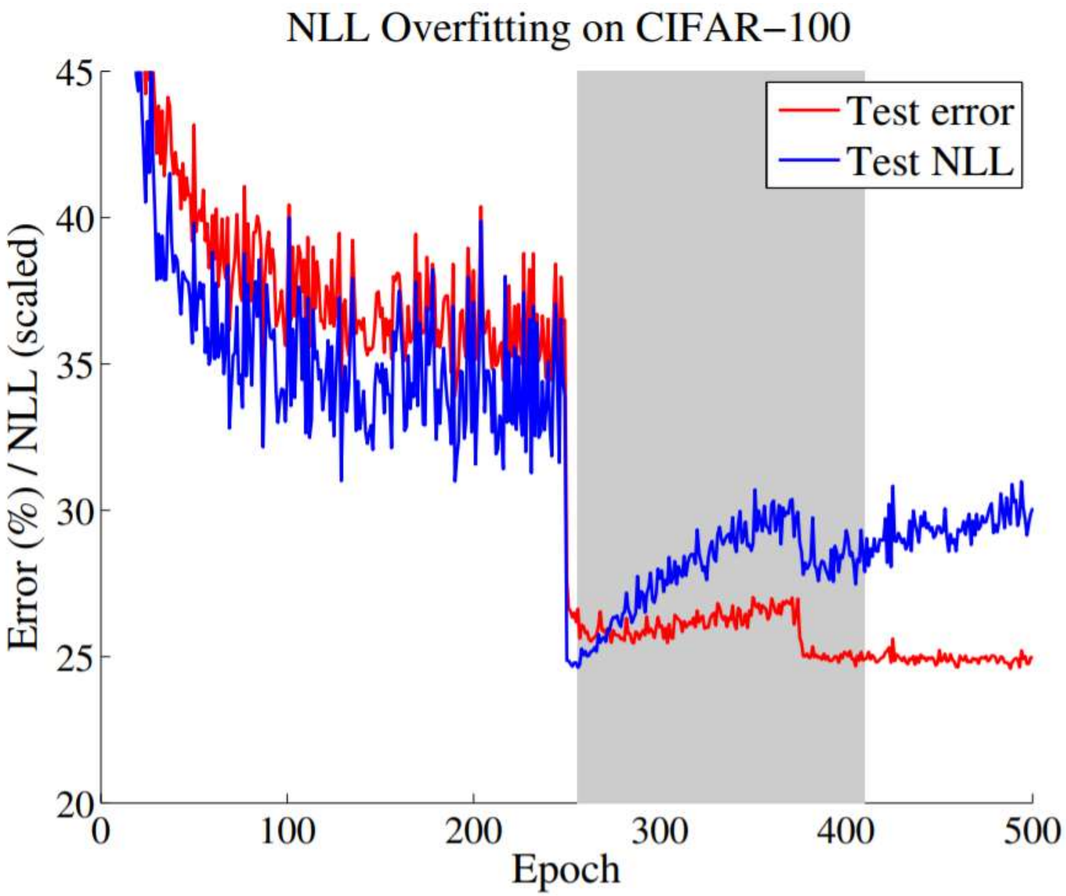


- У более точных сетей хуже калиброваны вероятности!

Калиброванность вероятностей

[Guo et al., 2017]

- Противоречие между точностью и правдоподобием



Как улучшить калиброванность?

[Guo et al., 2017]

- Простой рецепт – температура в softmax

$$p(y|x,\theta) = \frac{\exp \frac{1}{T} f_y(x,\theta)}{\sum_{s=1}^{K} \exp \frac{1}{T} f_s(x,\theta)}$$

- Параметр T надо настраивать отдельно после обучения

- По нему можно считать градиент, но стох. оптимизация работает плохо

- Способ работает лучше многих более сложных подходов

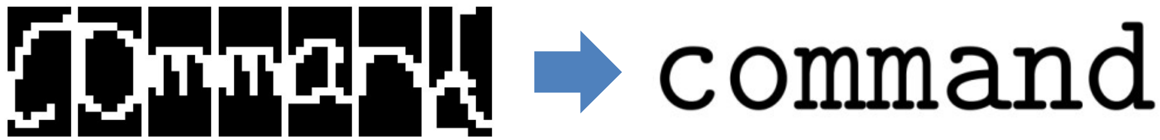
Как строить распределения сложных объектов?

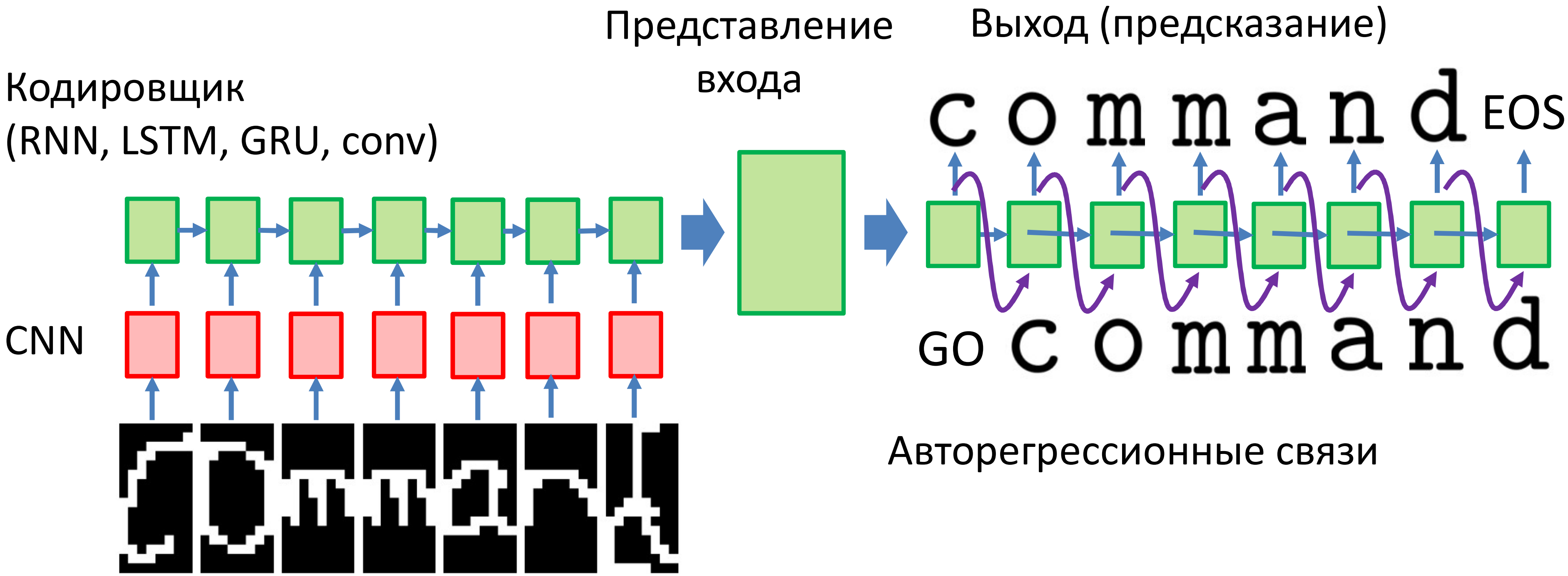
- Идея – использовать произведение условных вероятностей

$$P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = P(y_1 \mid \boldsymbol{x}, \boldsymbol{\theta}) P(y_2 \mid y_1, \boldsymbol{x}, \boldsymbol{\theta}) P(y_3 \mid y_2, y_1, \boldsymbol{x}, \boldsymbol{\theta}) \dots$$

- Упорядочить переменные и обрабатывать по очереди
- Авторегрессионные модели
- Идея появилась в модели NADE [Larochelle&Murray, 2011]
- Активно используется: seq2seq, PixelCNN, ByteNet, WaveNet
- В 2016-2018 это лучшие генеративные модели!

Последовательное предсказание

- Пример: 



Если не фиксирована длина выхода, то используют символ EOS

Как строить распределения сложных объектов?

- Идея – использовать произведение условных вероятностей

$$P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = P(y_1 \mid \boldsymbol{x}, \boldsymbol{\theta}) P(y_2 \mid y_1, \boldsymbol{x}, \boldsymbol{\theta}) P(y_3 \mid y_2, y_1, \boldsymbol{x}, \boldsymbol{\theta}) \dots$$

- Упорядочить переменные и обрабатывать по очереди

- Как параметризовать условные распределения?

- Варианты:
  - RNN
  - Masked CNN
  - Self-attention (transformer)

Условные распределения: RNN

- Идея – использовать произведение условных вероятностей

$$P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = P(y_1 \mid \boldsymbol{x}, \boldsymbol{\theta}) P(y_2 \mid y_1, \boldsymbol{x}, \boldsymbol{\theta}) P(y_3 \mid y_2, y_1, \boldsymbol{x}, \boldsymbol{\theta}) \dots$$

- Упорядочить переменные и обрабатывать по очереди

- RNN (LSTM, GRU, etc.)

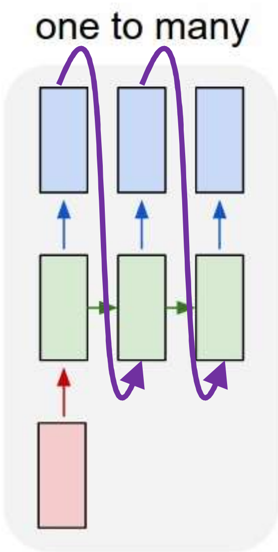


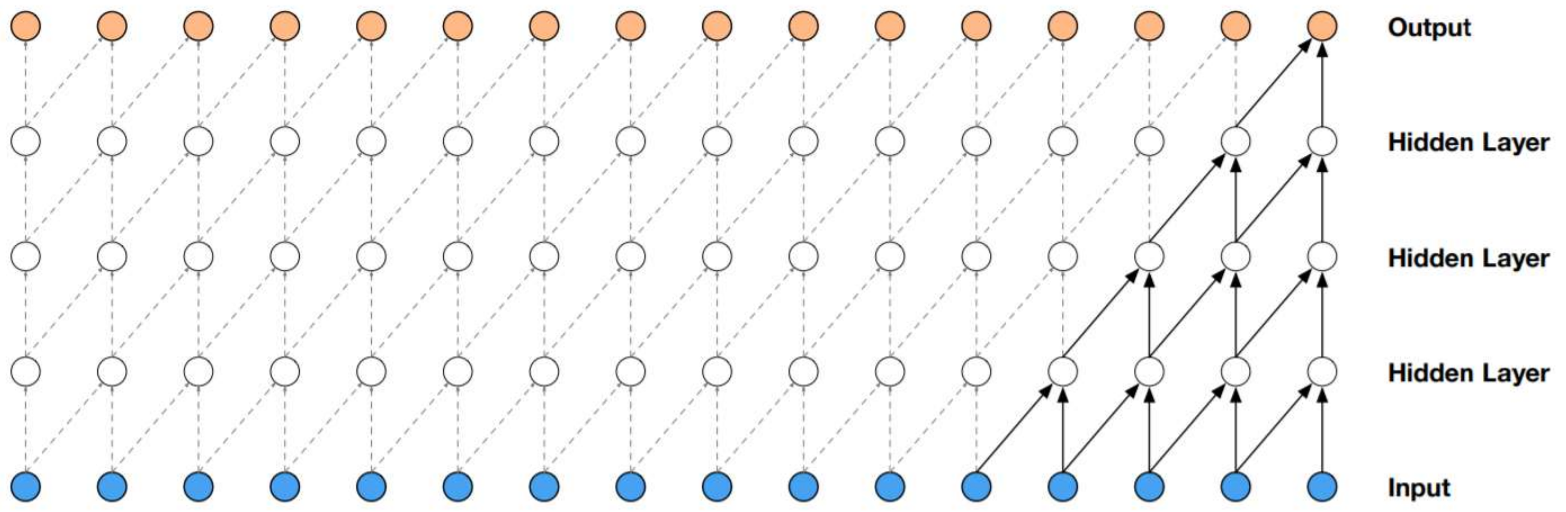
image credit: [Andrej Karpathy]

Условные распределения: Masked CNN

- Идея – использовать произведение условных вероятностей

$$P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = P(y_1 \mid \boldsymbol{x}, \boldsymbol{\theta}) P(y_2 \mid y_1, \boldsymbol{x}, \boldsymbol{\theta}) P(y_3 \mid y_2, y_1, \boldsymbol{x}, \boldsymbol{\theta}) \ldots$$

- Упорядочить переменные и обрабатывать по очереди
- Masked CNN (causal convolutions)
  – Медленно распространяется информация



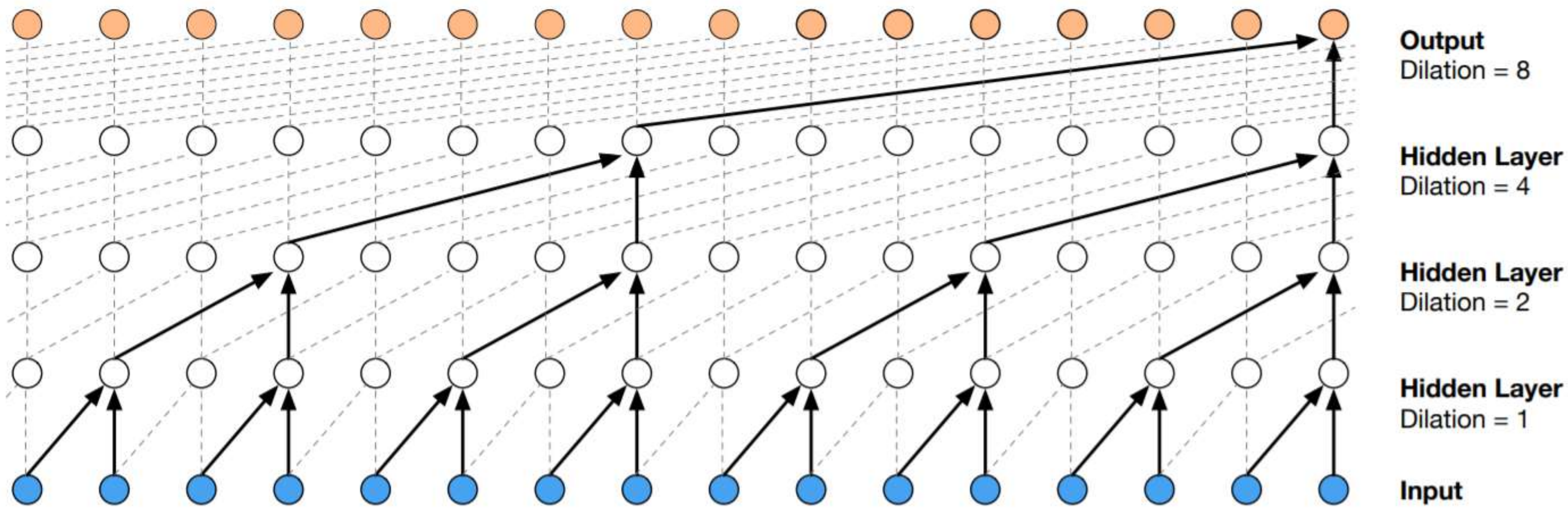
[van den Oord et al., 2016]

Условные распределения: Masked CNN

- Идея – использовать произведение условных вероятностей

$$P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = P(y_1 \mid \boldsymbol{x}, \boldsymbol{\theta}) P(y_2 \mid y_1, \boldsymbol{x}, \boldsymbol{\theta}) P(y_3 \mid y_2, y_1, \boldsymbol{x}, \boldsymbol{\theta}) \ldots$$

- Упорядочить переменные и обрабатывать по очереди
- Masked CNN (causal convolutions) + dilation
  – Информация распространяется экспоненциально быстрее



[van den Oord et al., 2016]

Условные распределения: Masked CNN

- Идея – использовать произведение условных вероятностей

$$P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = P(y_1 \mid \boldsymbol{x}, \boldsymbol{\theta}) P(y_2 \mid y_1, \boldsymbol{x}, \boldsymbol{\theta}) P(y_3 \mid y_2, y_1, \boldsymbol{x}, \boldsymbol{\theta}) \dots$$

- Упорядочить переменные и обрабатывать по очереди
- Masked CNN (causal convolutions) + dilation
  – Информация распространяется экспоненциально быстрее

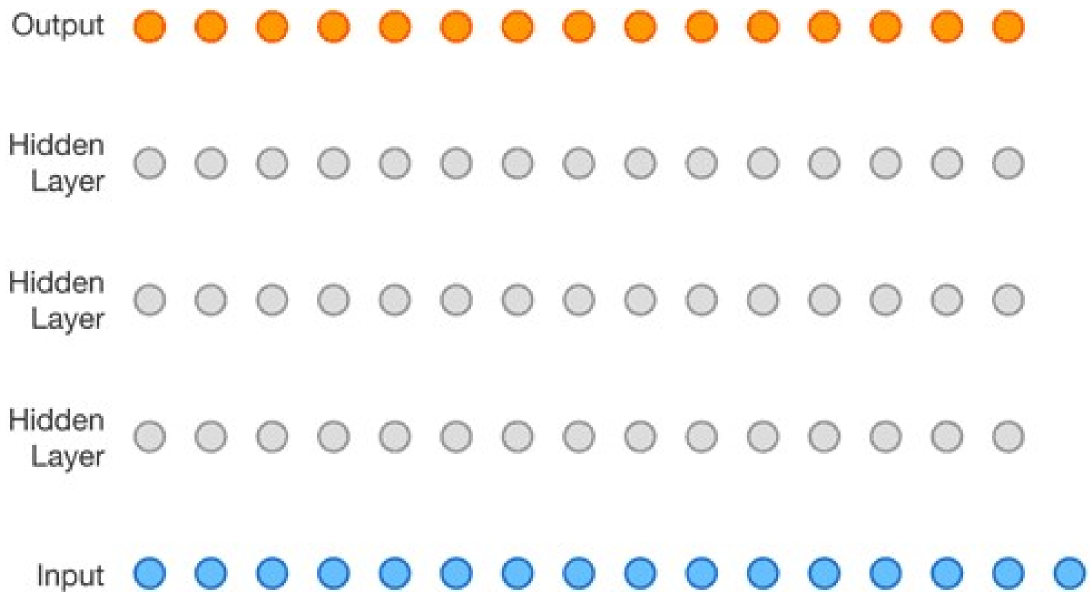


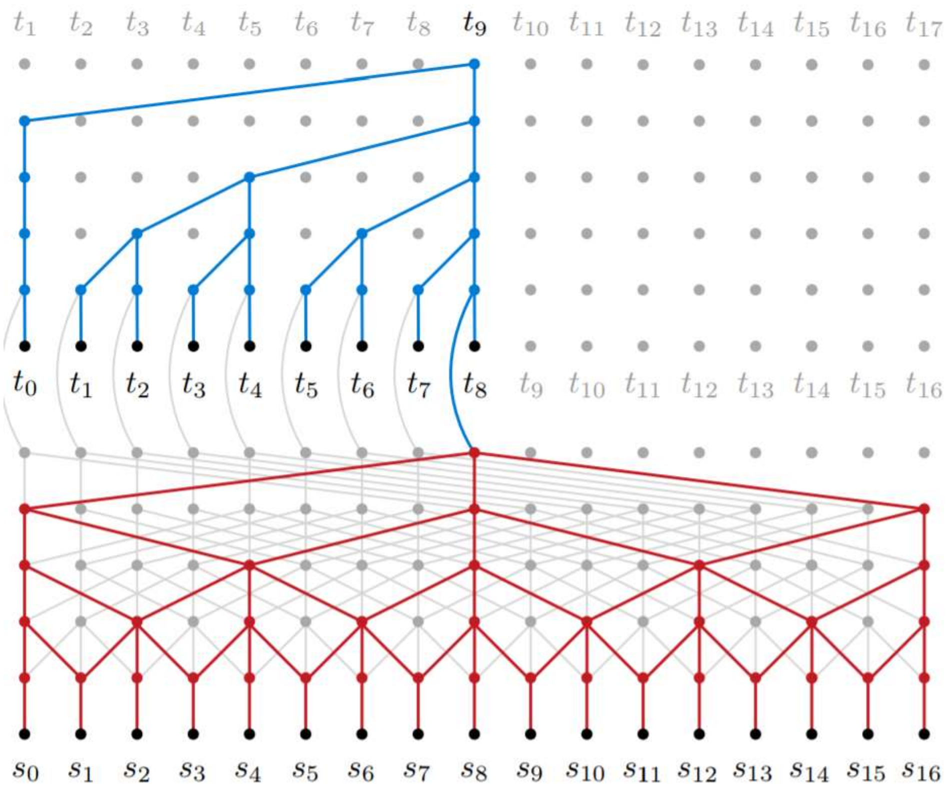
Image credit: https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/

Условные распределения: Masked CNN

- Идея – использовать произведение условных вероятностей

$$P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = P(y_1 \mid \boldsymbol{x}, \boldsymbol{\theta}) P(y_2 \mid y_1, \boldsymbol{x}, \boldsymbol{\theta}) P(y_3 \mid y_2, y_1, \boldsymbol{x}, \boldsymbol{\theta}) \ldots$$

- Упорядочить переменные и обрабатывать по очереди
- Encoder-decoder на основе Masked CNN + dilation



ByteNet [Kalchbrenner et al., 2016]

Условные распределения: Masked CNN

- Идея – использовать произведение условных вероятностей

$$P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = P(y_1 \mid \boldsymbol{x}, \boldsymbol{\theta}) P(y_2 \mid y_1, \boldsymbol{x}, \boldsymbol{\theta}) P(y_3 \mid y_2, y_1, \boldsymbol{x}, \boldsymbol{\theta}) \dots$$

- Упорядочить переменные и обрабатывать по очереди
- Masked CNN (causal convolutions) + dilation
  – Недостаток – медленная генерация (все последовательно)
  – Решение – дистилляция
  – Обучить параллельную модель на основе не параллельной

[van den Oord et al., 2017]

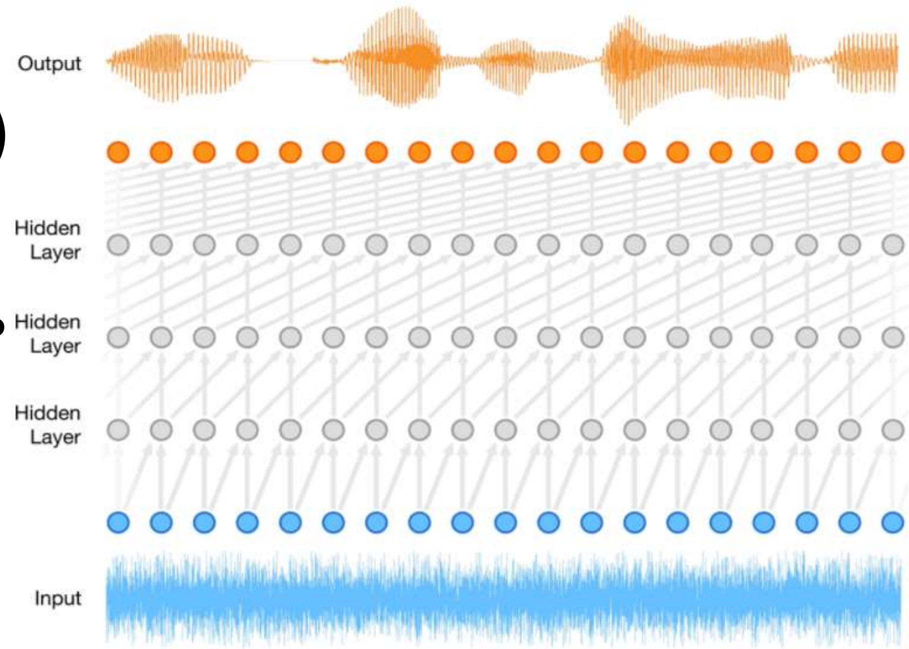


Image credit: https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/

Как генерировать изображения? PixelCNN

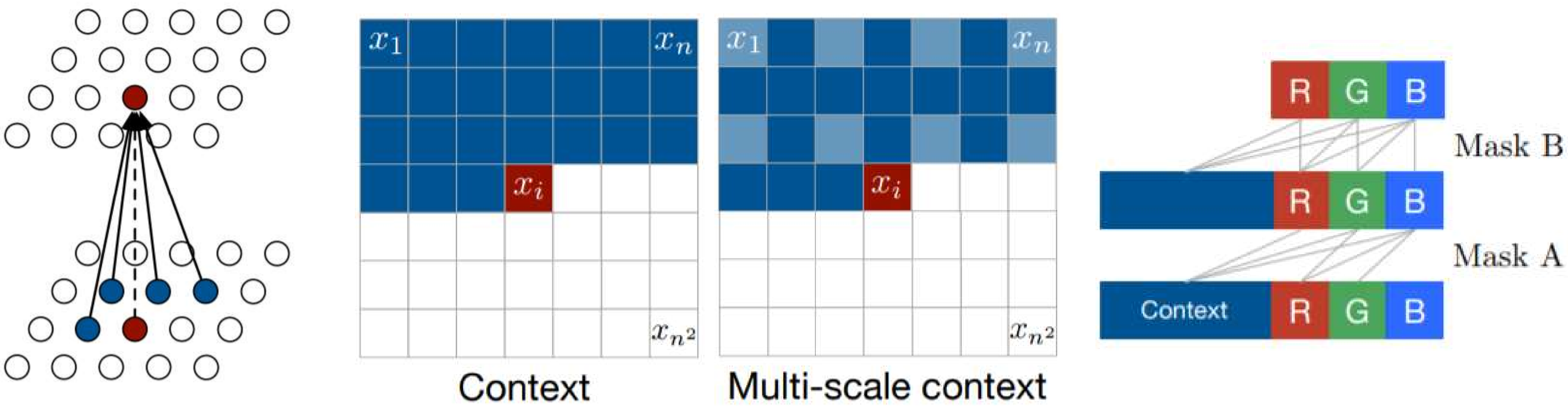
[van den Oord et al., 2016]

- Идея – использовать произведение условных вероятностей

$$P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = P(y_1 \mid \boldsymbol{x}, \boldsymbol{\theta}) P(y_2 \mid y_1, \boldsymbol{x}, \boldsymbol{\theta}) P(y_3 \mid y_2, y_1, \boldsymbol{x}, \boldsymbol{\theta}) \ldots$$

- Упорядочить переменные и обрабатывать по очереди

- Упорядочить пиксели!
- 2D causal convolutions



- Нюансы архитектуры: residual, batchnorm, etc.

Generative Adversarial Networks

[Goodfellow et al., 2014]

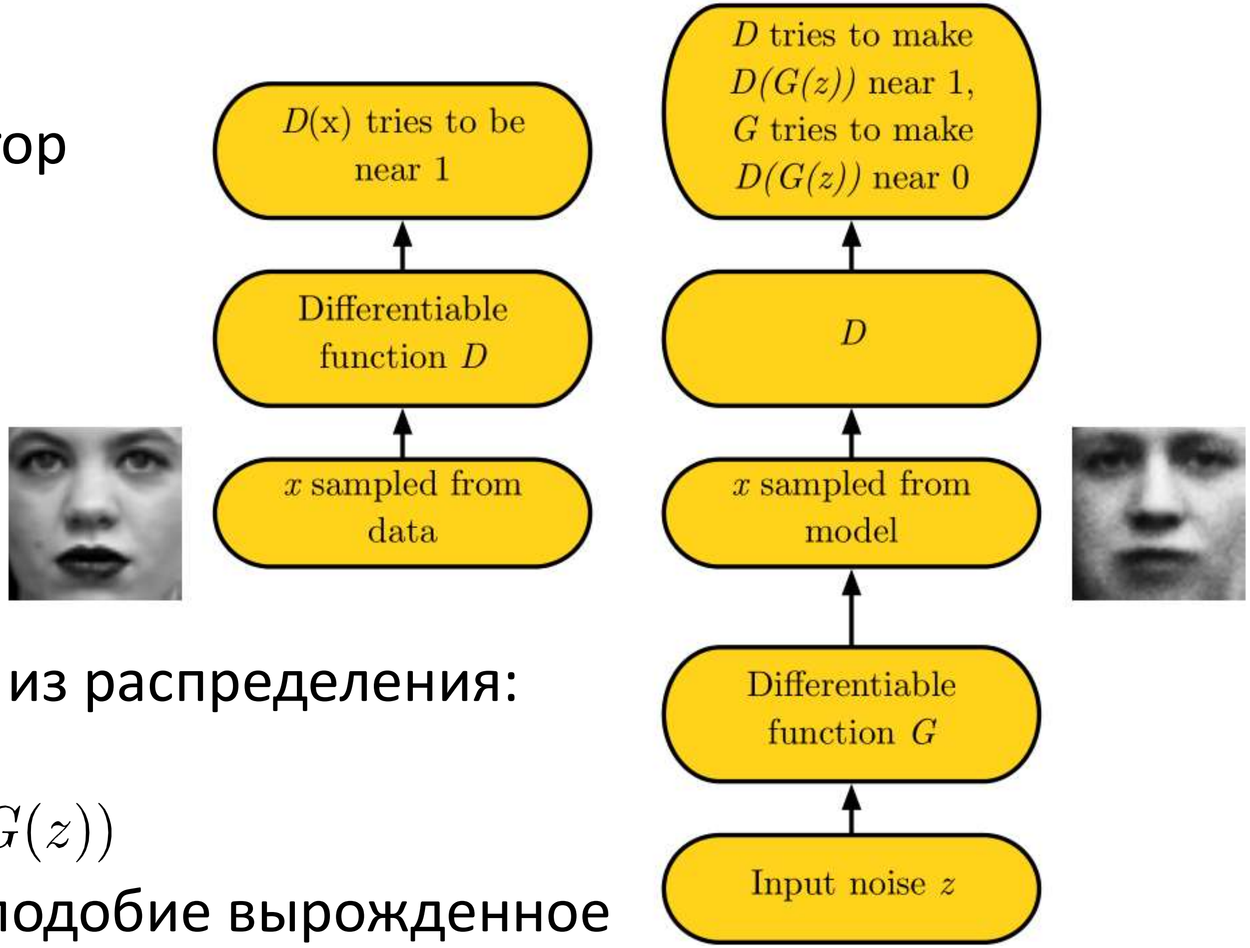
- Генеративные модели (обычно для изображений)
- Основная идея: вместо определения целевой функции (правдоподобие, ошибка реконструкции)
  целевая функция обучается вместе с данными
- Генератор – сеть, синтезирующая картинки из шума
- Дискриминатор – сеть, отличающая настоящие от синтезированных

Generative Adversarial Networks

[Goodfellow et al., 2014]

Две сети:

- G – генератор
- D – дискриминатор



- GAN генерируют из распределения:

$$z \sim p_z(z)$$

$$x = G(z) \sim \delta(x = G(z))$$

- Полного правдоподобие вырожденное

$$p(x, z) = \delta(x = G(z)p_z(z)$$

- Неявные (implicit) модели

Image credit: Ian Goodfellow

Generative Adversarial Networks

[Goodfellow et al., 2014]

- GAN генерируют из распределения:

$$z \sim p_z(z)$$

$$x = G(z) \sim p_m = \delta(x = G(z))$$

- Как обучаются GANы?
  – Поиск седловой точки стох. оптимизацией

$$\min_G \max_D \mathbb{E}_{x \sim p_d}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

- Как это связано с распределениями? [Nowozin et al., 2016]
  – f-дивергенции между распределениями

$$D_f(p_d \| p_m) = \int p_m(x) f\left(\frac{p_d(x)}{p_m(x)}\right) dx \qquad \mathrm{KL}: f(u) = u \log(u)$$

  – Двойственной представление дивергенции

$$D_f(p_d \| p_m) = \sup_T \left(\mathbb{E}_{x \sim p_d(x)}[T(x)] - \mathbb{E}_{x \sim p_m(x)}[f^*(T(x))]\right)$$

  – f = Jensen-Shannon divergence и некоторая T дают GAN

Generative Adversarial Networks

[Goodfellow et al., 2014]

- GAN генерируют из распределения:

$$z \sim p_z(z)$$

$$x = G(z) \sim p_m = \delta(x = G(z))$$

- Как обучаются GANы?
  – Поиск седловой точки стох. оптимизацией

$$\min_G \max_D \mathbb{E}_{x \sim p_d}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))]$$

- GANы не минимизируют дивергенцию в явном виде
  – На внутреннем шаг нет полной оптимизации (1-5 шагов)
  – Supremum не по всем функциям, а по параметрам нейросети

- Можно использовать и другие «близости» распределений
  – Wasserstein GAN, MMD GAN, Cramer GAN, etc.

Variational auto-encoders

[Kingma&Welling, 2014]

- Рассмотрим вероятностную модель
  $$z \sim p_z(z)$$
  $$x \sim p(x|G(z)) = \mathcal{N}(\mu_G(z), \sigma_G^2(z))$$
- Можно вычислить полное правдоподобие
- Как настраивать *G*?
  $$p(x, z|G) = p(x|G(z))p_z(z)$$
- Метод. макс. правдоподобия?
  – Правдоподобие: $p(x|G) = \boxed{\int p(x|G(z))p_z(z)dz}$
  – Если *G* – линейная функция, то интеграл берётся
  – Вероятностный PCA (метод главных компонент)
  – Если *G* сложнее, то интеграл не берётся!
- EM-алгоритм?
  – E-шаг: апостериорное распр. $q(z|x) = p(z|x, G) = \frac{p(x|G(z))p_z(z)}{\boxed{\int p(x|G(z))p_z(z)dz}}$
  – M-шаг: $\max_G \mathbb{E}_{z \sim q(z|x)} p(x, z|G)$

  Интеграл не берётся!
- Приближенный вывод!

Variational auto-encoders

[Kingma&Welling, 2014]

- Рассмотрим вероятностную модель

$$z \sim p_z(z)$$

$$x \sim p(x|G(z)) = \mathcal{N}(\mu_G(z), \sigma_G^2(z)) \qquad q(z|x, D) = \mathcal{N}(\mu_D(x), \sigma_D^2(x))$$

- Приближенный вывод с вариационной нижней оценкой!

$$\log p(x|G) = \log \int p(x|G(z))p_z(z)dz = \log \int p(x|G(z))p_z(z)\frac{q(z|x,D)}{q(z|x,D)}dz$$

$$= \log \mathbb{E}_{z\sim q(z|x,D)} \frac{p(x|G(z))p_z(z)}{q(z|x,D)}dz$$

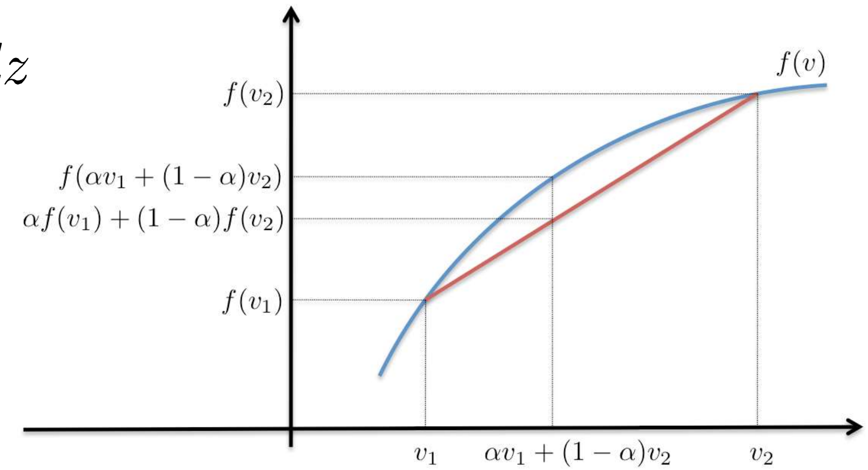
$$\geq \mathbb{E}_{z\sim q(z|x,D)} \log \frac{p(x|G(z))p_z(z)}{q(z|x,D)}dz$$

$$= \mathrm{ELBO}(x, G, D)$$

$$= \mathbb{E}_{z\sim q(z|x,D)} \log p(x|G(z))$$

$$- \mathrm{KL}(q(z|x,D)\|p_z(z))$$

Jensen's inequliaty!



Variational auto-encoders

[Kingma&Welling, 2014]

- Рассмотрим вероятностную модель

$$z \sim p_z(z)$$

$$x \sim p(x|G(z)) = \mathcal{N}(\mu_G(z), \sigma_G^2(z)) \qquad q(z|x,D) = \mathcal{N}(\mu_D(x), \sigma_D^2(x))$$

- Приближенный вывод с вариационной нижней оценкой!

$$\mathrm{ELBO}(x,G,D) = \mathbb{E}_{z\sim q(z|x,D)} \log p(x|G(z)) - \mathrm{KL}(q(z|x,D)\|p_z(z))$$

- Будет максимизировать ELBO при помощи SGD!

- Стохастический градиент:

$$\nabla_G = \mathbb{E}_{z\sim q(z|x,D)} \nabla_G \log p(x|G(z)) \approx \nabla_G \log p(x|G(\tilde{z})),\ \tilde{z} \sim q(z|x,D)$$

Монте-Карло оценка!

$$\nabla_D = \underbrace{\nabla_D \mathbb{E}_{z\sim q(z|x,D)} \log p(x|G(z))} - \underbrace{\nabla_D \mathrm{KL}(q(z|x,D)\|p_z(z))}$$

Нельзя занести градиент внутрь!
Можно, но потеряем МО и Монте-Карло оценки

Считается аналитически!

Градиент по распределению?

[Kingma&Welling, 2014]

- Рассмотрим вероятностную модель

$$z \sim p_z(z)$$

$$x \sim p(x|G(z)) = \mathcal{N}(\mu_G(z), \sigma_G^2(z)) \qquad q(z|x, D) = \mathcal{N}(\mu_D(x), \sigma_D^2(x))$$

- Градиент ELBO по *D*

$$\nabla_D^{\text{data}} = \nabla_D \mathbb{E}_{z\sim q(z|x,D)} \log p(x|G(z)) = \int \nabla_D [q(z|x, D)] \log p(x|G(z)) dz$$

- Общий метод – log-derivative trick (REINFORCE)
  – Производная логарифма: $\nabla_D[\log q(z|x, D)] = \frac{\nabla_D[q(z|x,D)]}{q(z|x,D)}$

$$\nabla_D^{\text{data}} = \int \nabla_D[\log q(z|x, D)] q(z|x, D) \log p(x|G(z)) dz$$
$$= \mathbb{E}_{z\sim q(z|x,D)} \nabla_D [\log q(z|x, D)] \log p(x|G(z))$$
$$\approx \nabla_D[\log q(\tilde{z}|x, D)] \log p(x|G(z)), \;\; \tilde{z} \sim q(z|x, D)$$

Числа порядка -100, -1000

Небольшие числа обоих знаков

- Задача решена?
  – Очень большая дисперсия градиента!

Градиент по распределению?

[Kingma&Welling, 2014]

- Рассмотрим вероятностную модель

$$z \sim p_z(z)$$

$$x \sim p(x|G(z)) = \mathcal{N}(\mu_G(z), \sigma_G^2(z)) \qquad q(z|x, D) = \mathcal{N}(\mu_D(x), \sigma_D^2(x))$$

- Градиент ELBO по *D*

$$\nabla_D^{\text{data}} = \nabla_D \mathbb{E}_{z \sim q(z|x,D)} \log p(x|G(z))$$

- У оценки очень большая дисперсия!
- В RL много методов понижения дисперсии (baselines, etc.)
- Репараметризация (если возможна) – самое лучшее!
  – Разделение случайности и параметров
  – Представим распределение *q(z|x, D)* как $g(x, D, \varepsilon), \ \varepsilon \sim r(\varepsilon)$

$$z = \mu_D(x) + \sigma_D(x)\varepsilon, \ \varepsilon \sim r(\varepsilon)$$

g – детерминированная функция

ε – шум

  – Тогда градиент легко оценить:

$$\nabla_D^{\text{data}} = \nabla_D \int q(z|x, D) \log p(x|G(z)) dz = \int r(\varepsilon) \nabla_D \big[\log p(x|G(g(x, D, \varepsilon))\big] d\varepsilon$$

Как работает VAE?

- Восстанавливает распределения!
- Но, размытые картинки ☹

GANы [Karras et al. 2017]

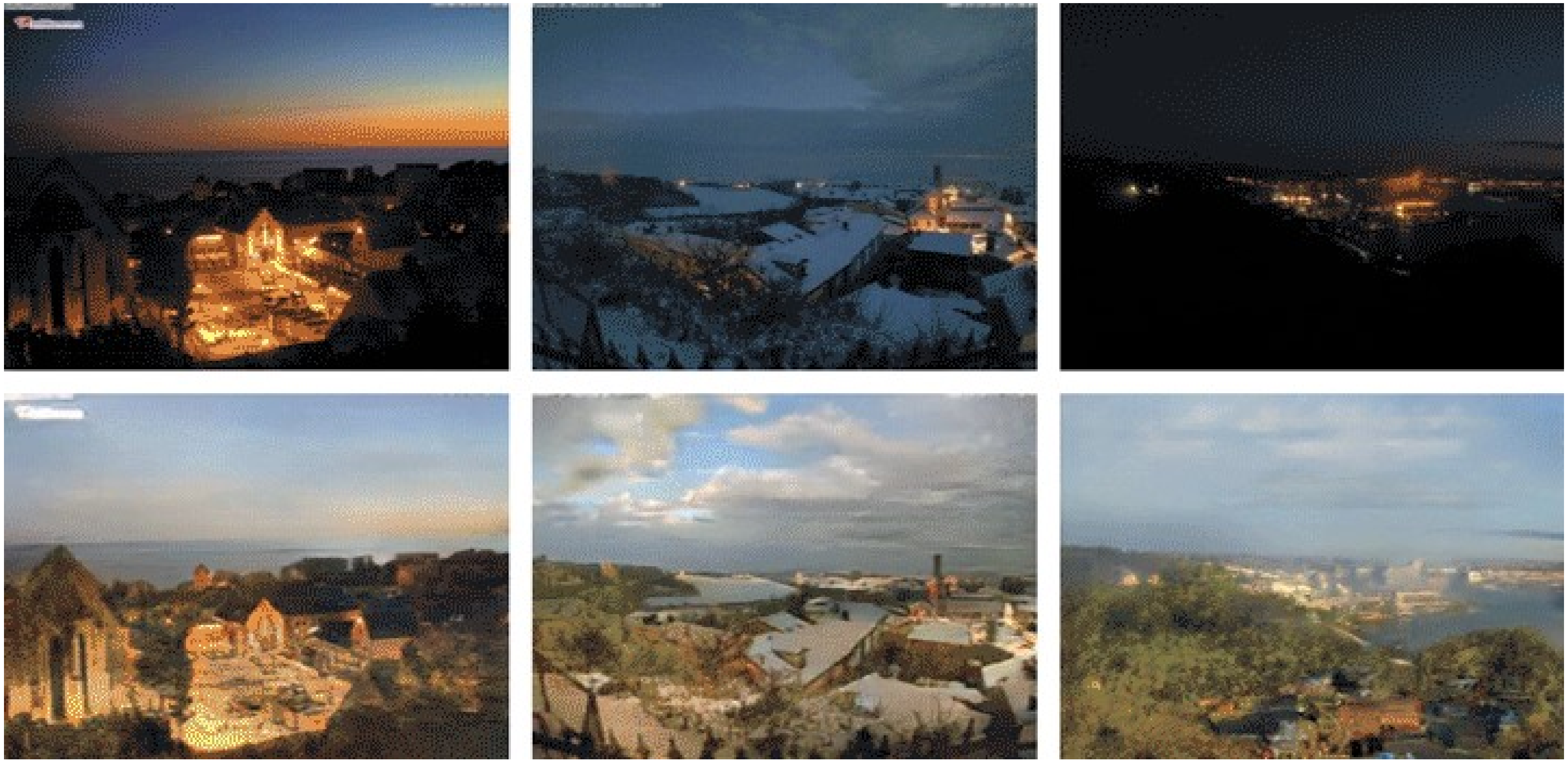
Image credit: Alec Radford

Комбинации VAE и GAN

- Можно восстанавливать красивые изображения + получать мульти-модальность

[Zhu et al., 2017]

- Мульти-модальный pix2pix



Заключение

- Авторегрессионные модели
  – генерируют хорошие самплы (особенно текст и звук)
  – нет скрытых представлений

- GAN
  – Могут генерировать красивые картинки
  – Проблемы с выучиванием распределений

- VAE
  – Хорошо восстанавливают распределения
  – Размытые картинки