Глубинное обучение

Лекция 10: Вероятностные модели и нейросети

Лектор: Максим Рябинин
Старший исследователь, Yandex Research

Программа ML Residency: yandex.ru/yaintern/research_ml_residency

ФКН ВШЭ, 2022

План лекции

- Нейросети для параметризации вероятностных распределений
- Авторегрессионные модели
  - Seq2seq, ByteNet, PixelCNN
- Неявные вероятностные модели
  - GAN
- Вероятностные модели со скрытыми переменными
  - VAE
- Продвинутые классы моделей
  - Normalizing Flows

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

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

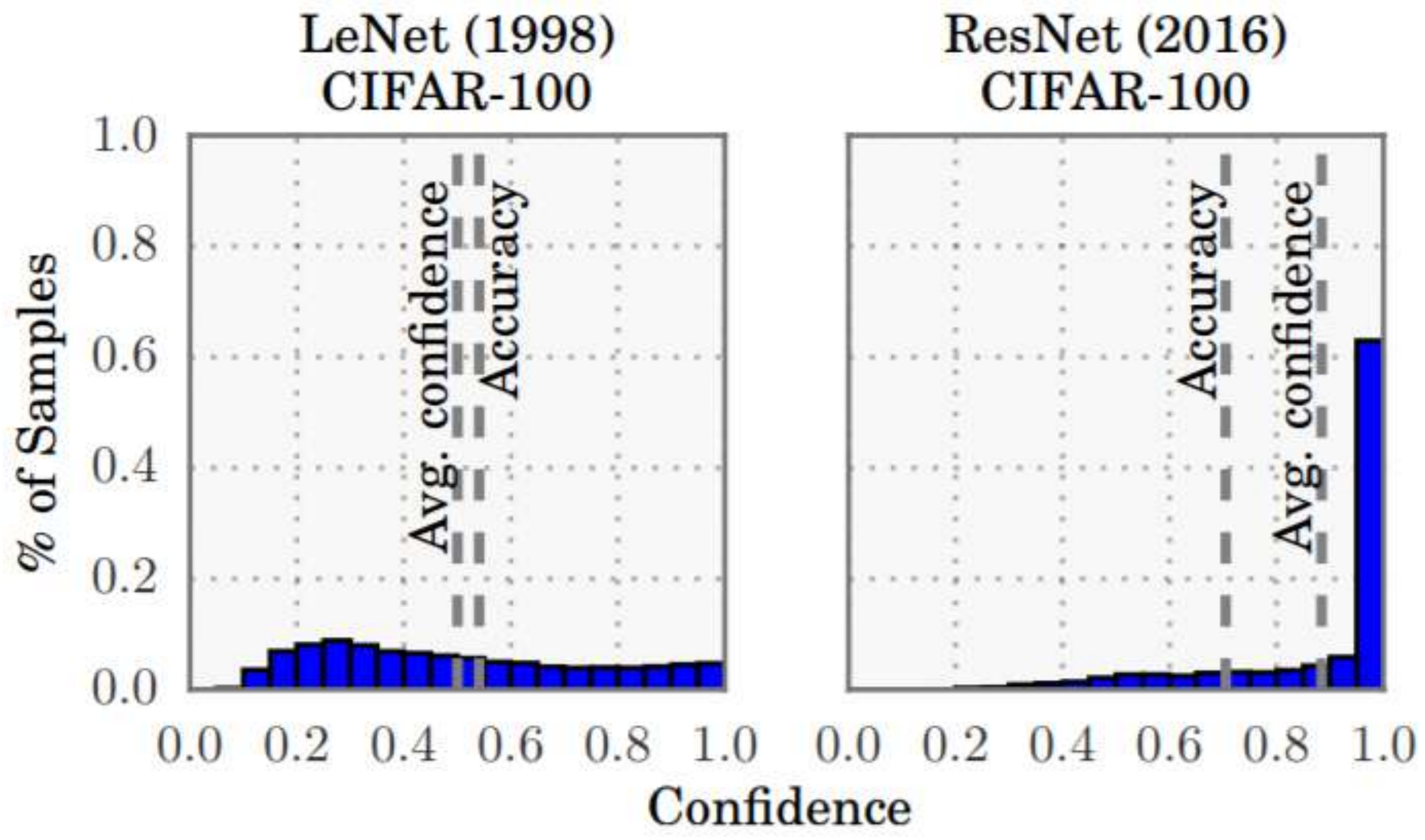
- Правдоподобие – вероятность правильных ответов
- Вероятности показывают «уверенность» сети
- Настоящие ли это вероятности?

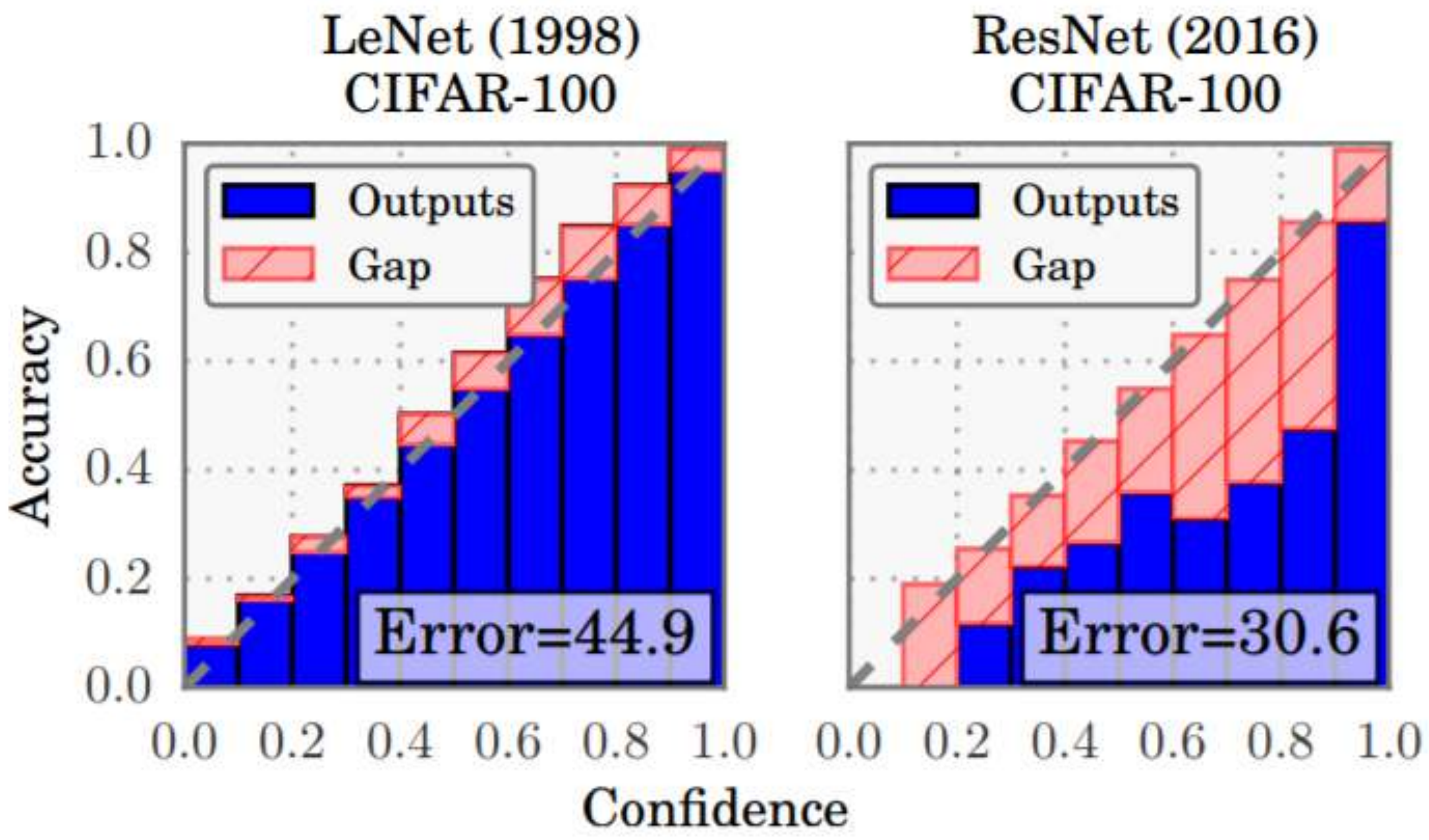
Калиброванность вероятностей

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

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

- Выдают ли нейросети калиброванные вероятности?

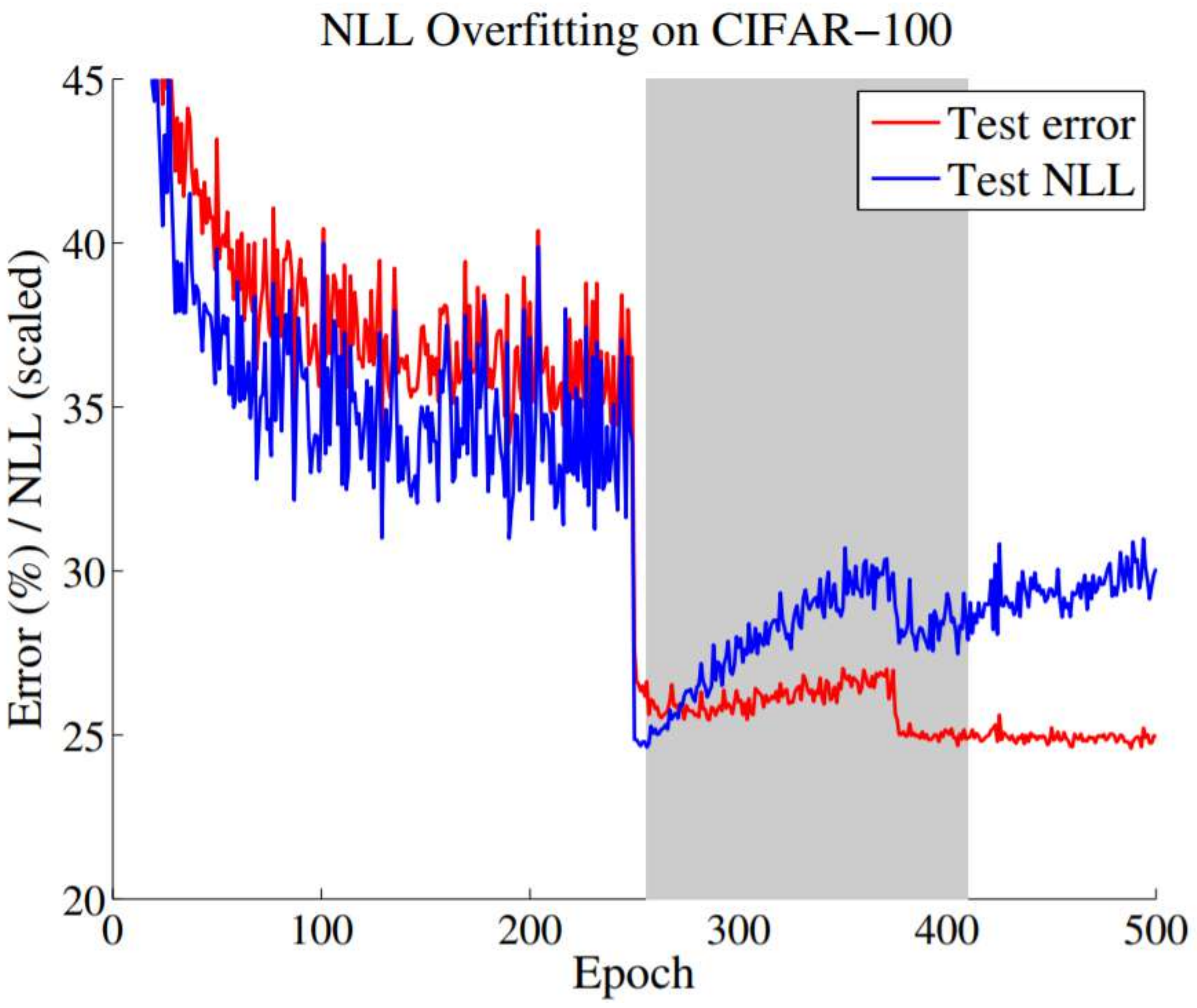




- У более точных моделей хуже калиброваны вероятности!

Калиброванность вероятностей

- Противоречие между точностью и правдоподобием



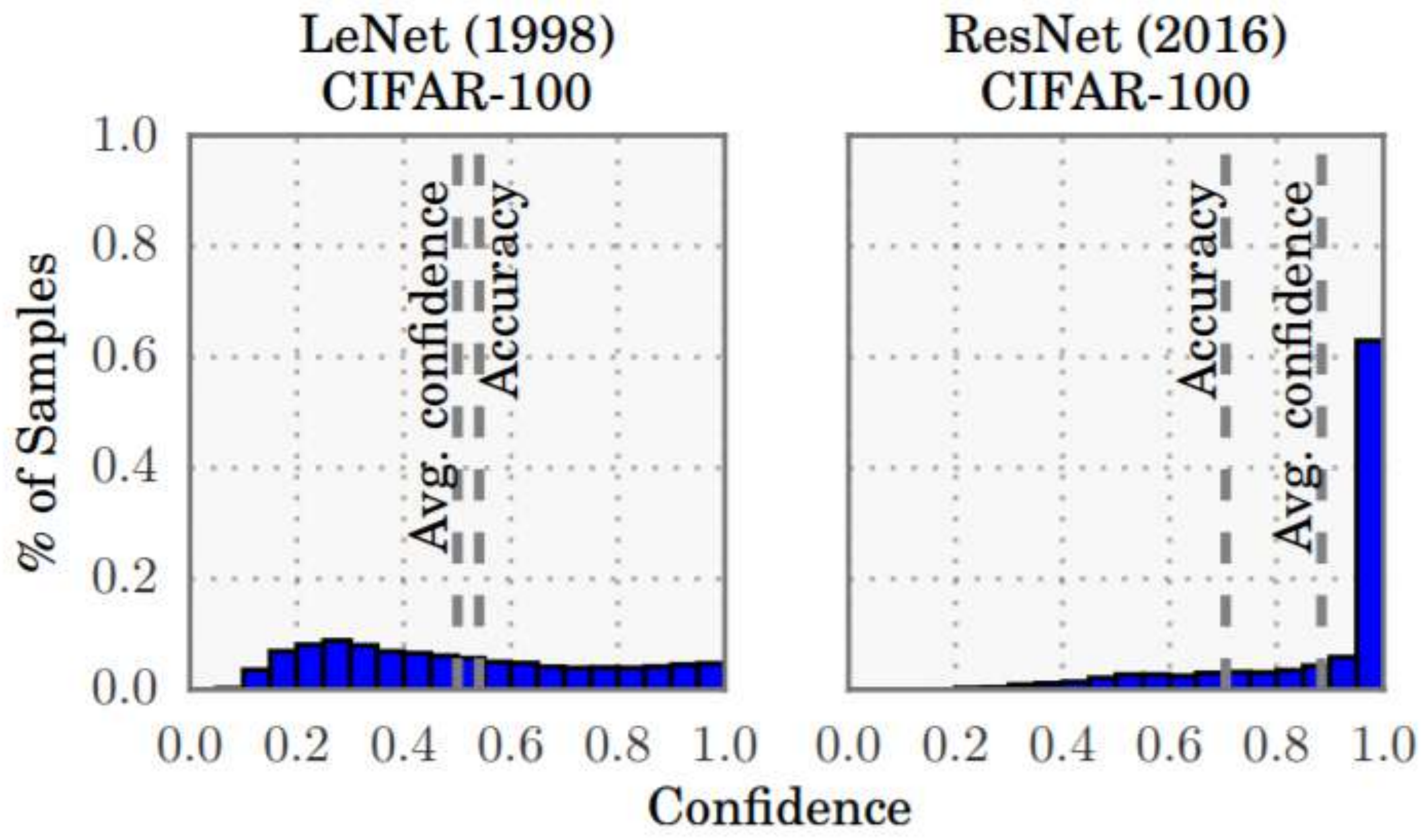
[Guo et al., 2017]
arXiv:1706.04599
[Minderer et al., 2021]
arXiv:2106.07998

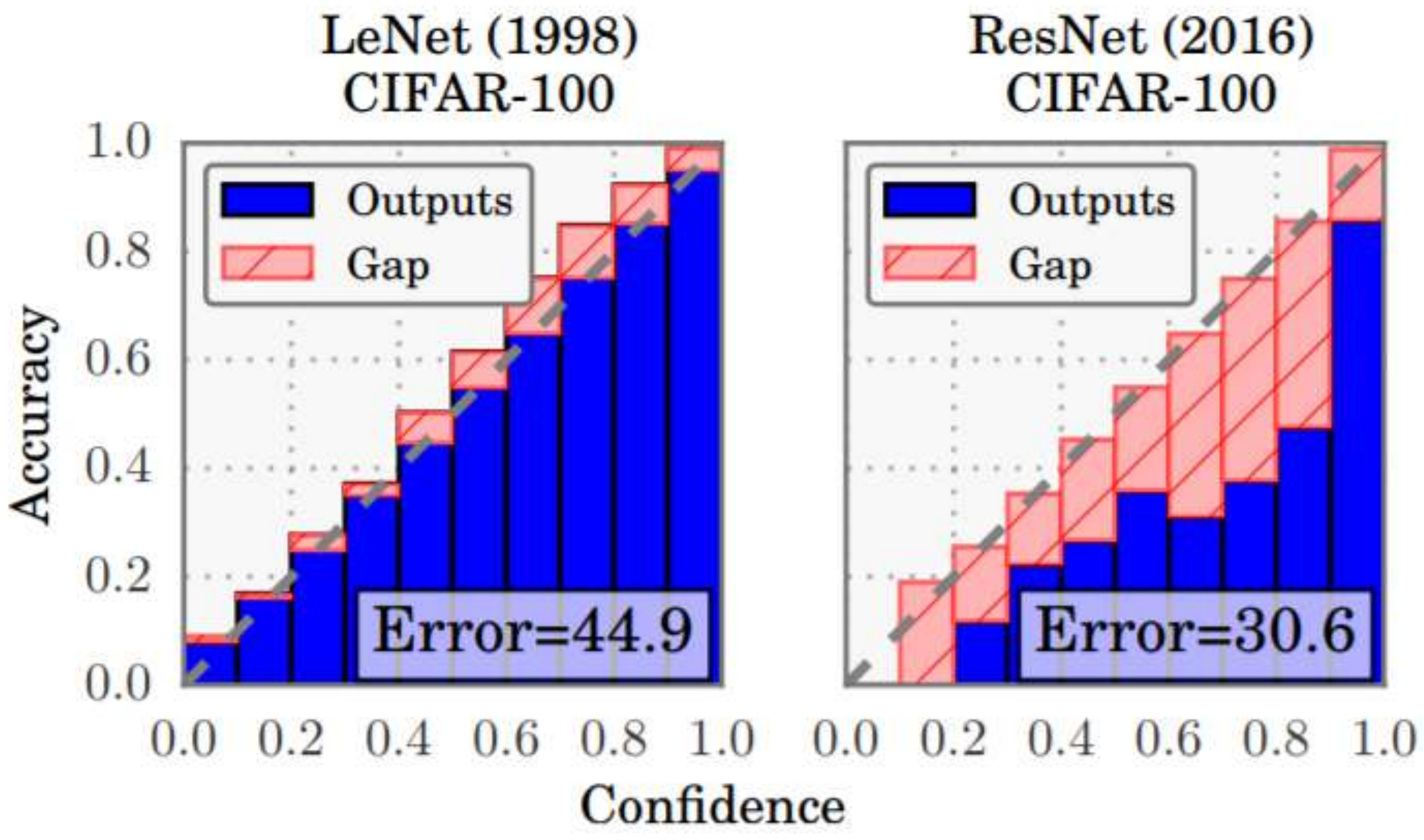
Калиброванность вероятностей

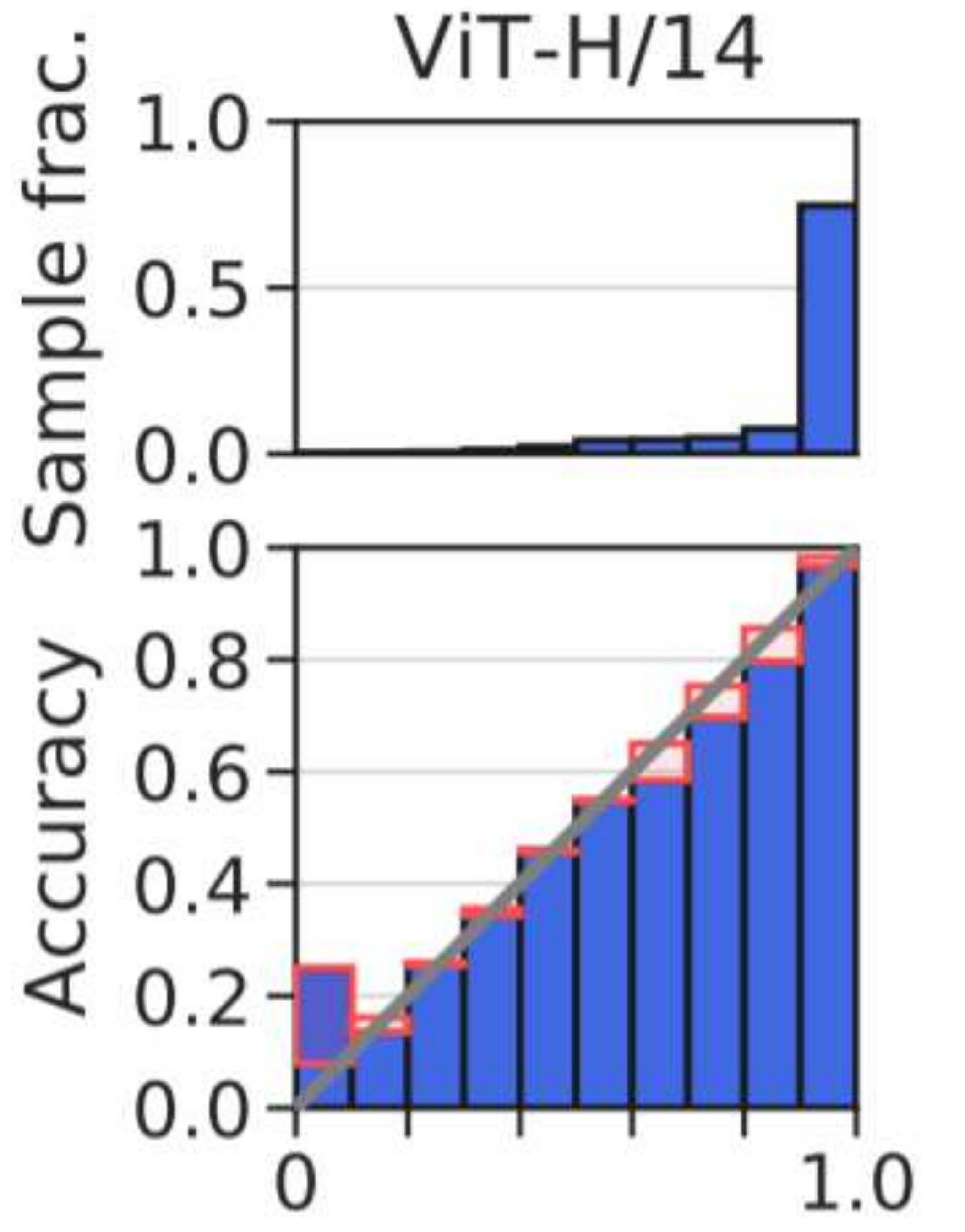
- Калиброванность – удобное свойство вероятностей
- Калиброванность означает, что если вероятность *p*, то ответ правильный в доле случаев *p*

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

- Выдают ли нейросети калиброванные вероятности?







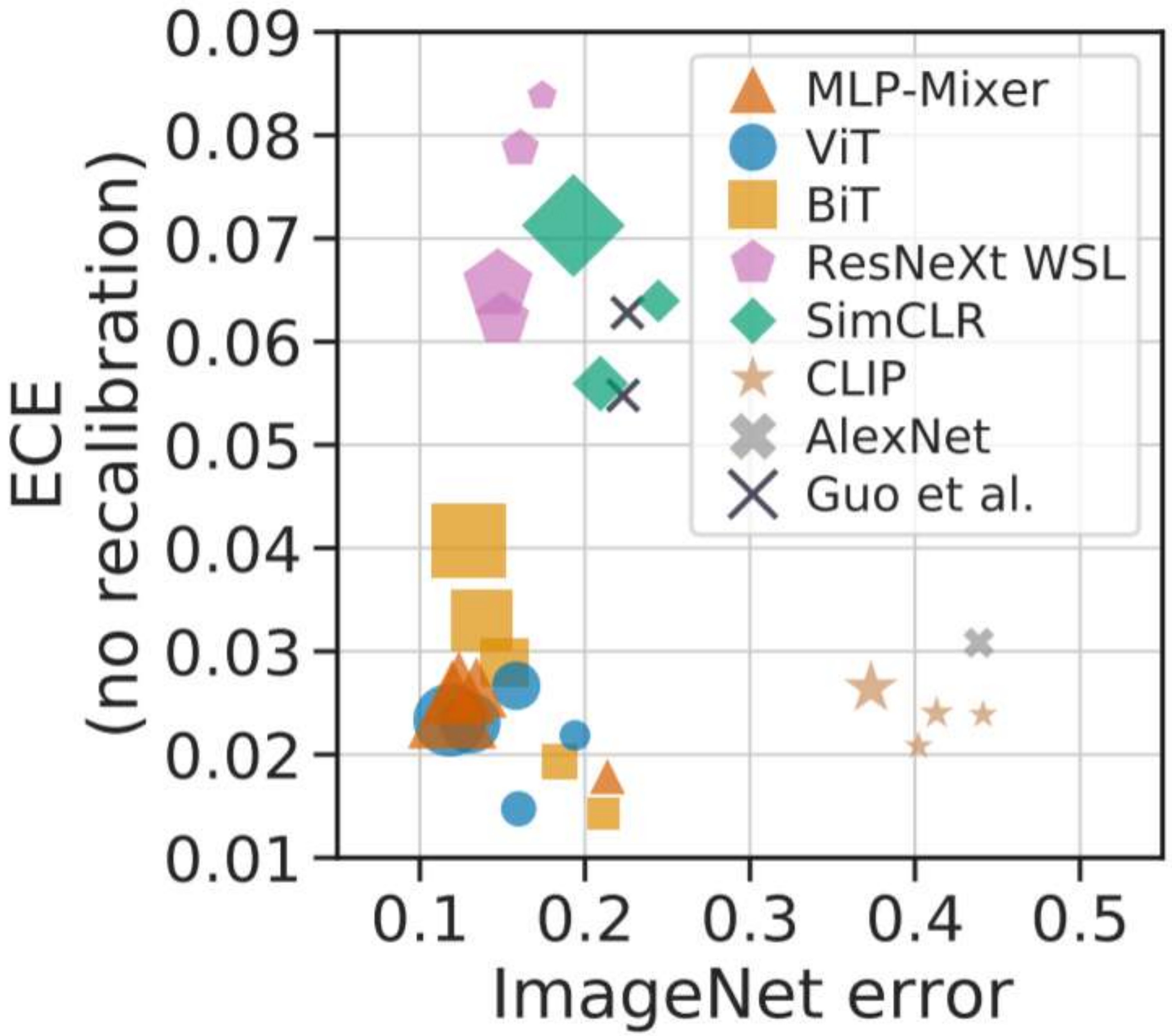
- У более точных моделей хуже калиброваны вероятности!

Не всегда!

Калиброванность вероятностей

- Архитектура модели – важный фактор

- Модели со свертками калиброваны хуже (много нюансов)



[Guo et al., 2017]
arXiv:1706.04599

Как улучшить калиброванность?

- Простой рецепт – температура в softmax

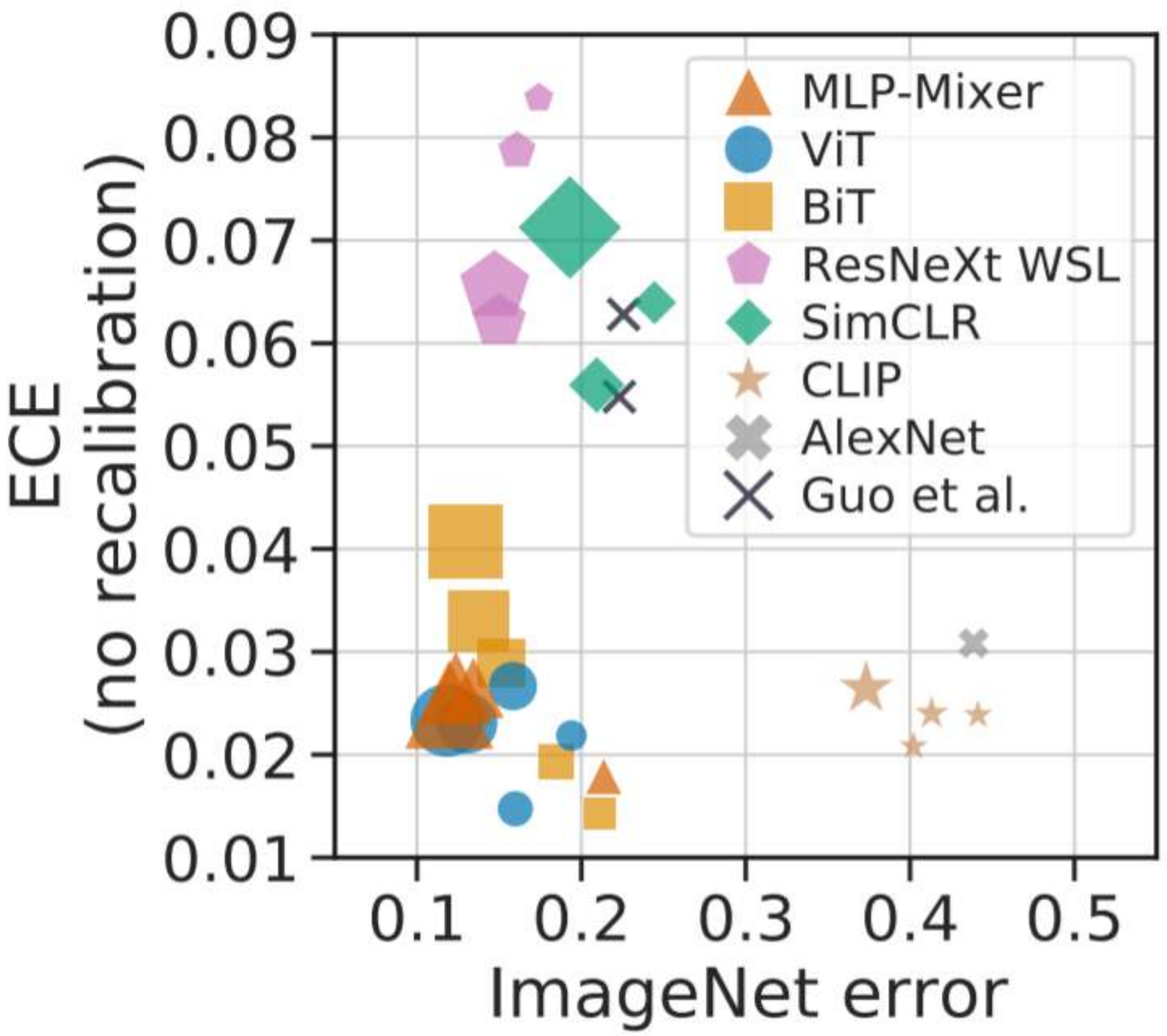
$$p(y|x,\theta) = \frac{\exp \frac{1}{T} f_y(x,\theta)}{\sum_{s=1}^{K} \exp \frac{1}{T} f_s(x,\theta)}$$

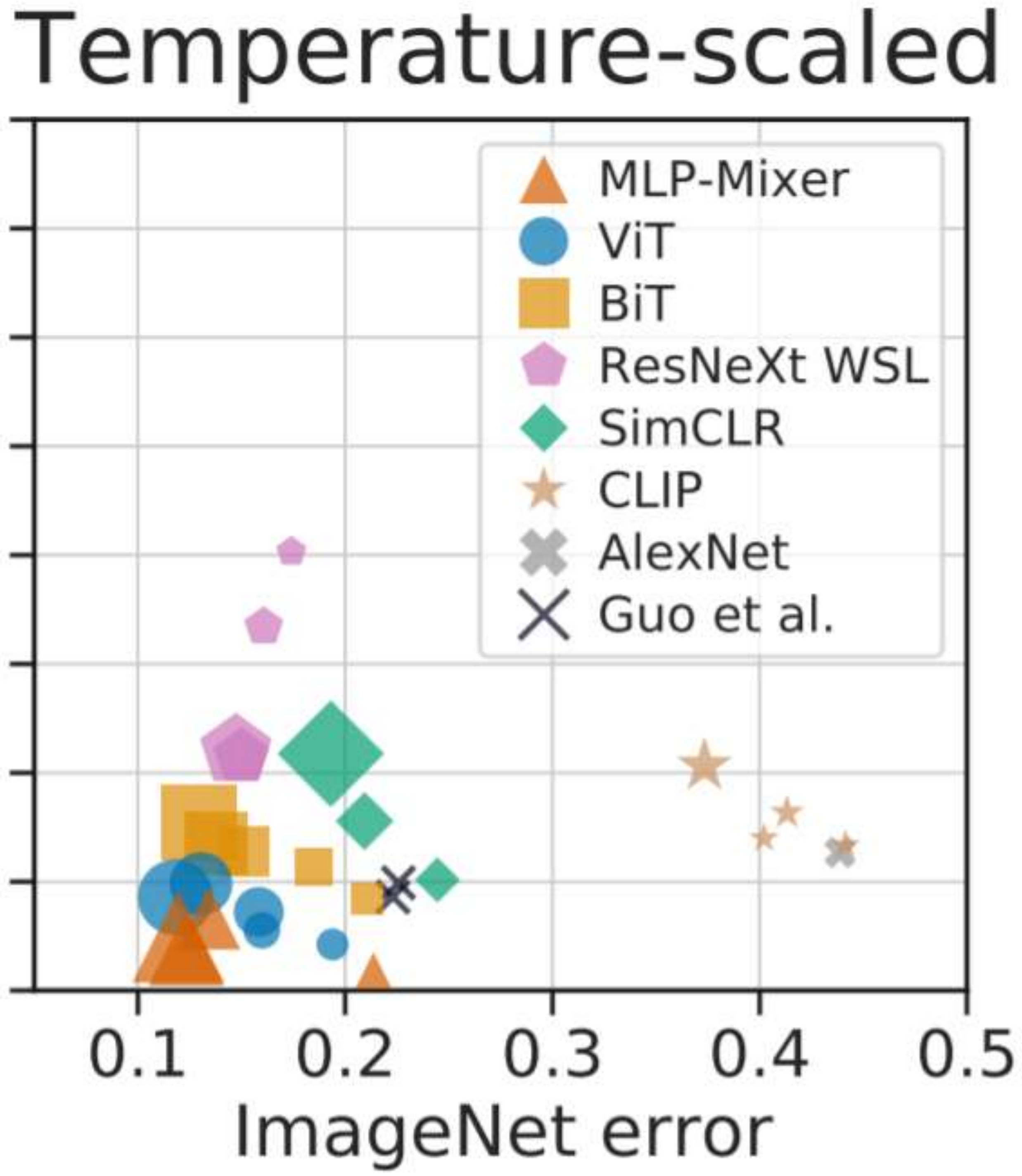
- Параметр T надо настраивать отдельно после обучения
- По нему можно считать градиент, но стох. оптимизация работает плохо
- Способ работает лучше многих более сложных подходов

[Minderer et al., 2021]
arXiv:2106.07998

Калиброванность вероятностей

- Архитектура модели – важный фактор
- Модели со свертками калиброваны хуже (много нюансов)
- Температура помогает!





Как улучшить калиброванность?

- 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 стохастическая оптимизация работает плохо
- Способ работает лучше многих более сложных подходов
- Еще лучше (но дороже) – ансамбли сетей (deep ensembles; Lakshminarayanan et al., 2017)
- Литература: Guo et al., 2017; arXiv:1706.04599]

  [Ashukha et al., 2020; arXiv:2002.06470]

  [Minderer et al., 2021; arXiv:2106.07998]

[Guo et al., 2017]
arXiv:1706.04599

Авторегрессионные модели

- 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
- Авторегрессионные модели
- Идея появилась в 90-х
- Активно используется: seq2seq, PixelCNN, ByteNet, WaveNet

Авторегрессионные модели

- 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.)

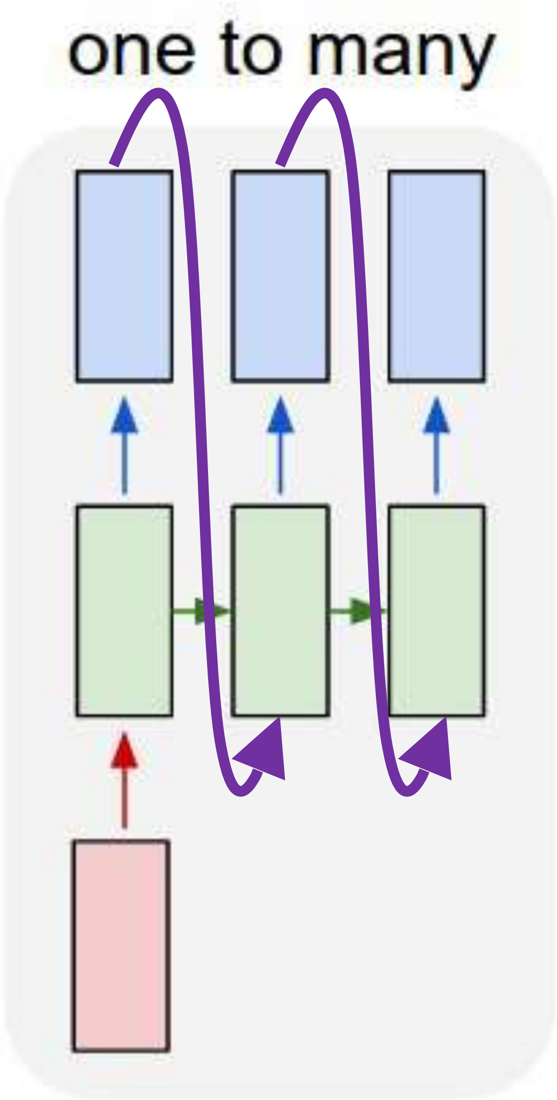


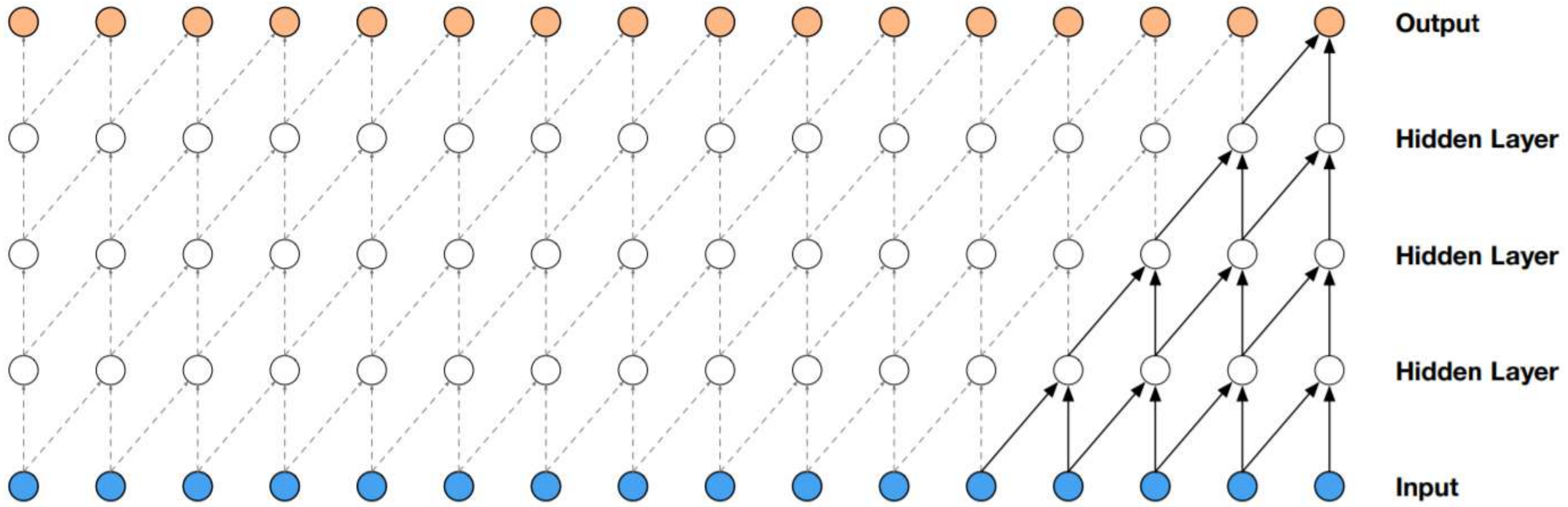
image credit: **Andrej Karpathy**

Условные распределения: Masked CNN

- Идея – использовать произведение условных вероятностей

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

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



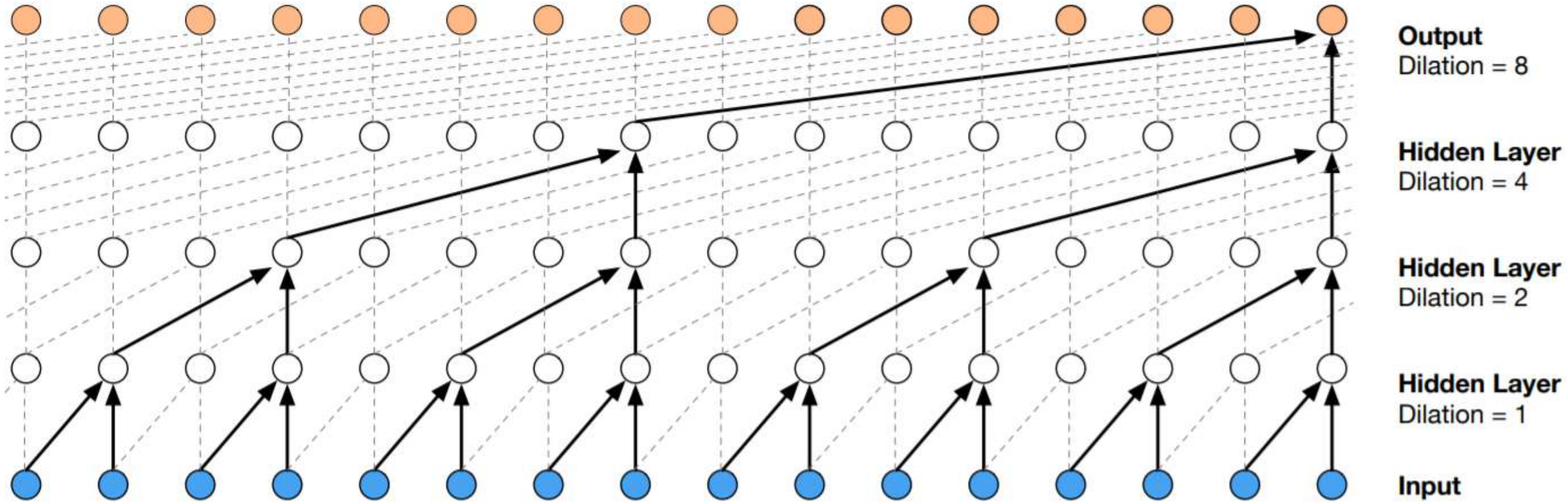
[van den Oord et al., 2016]

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

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

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

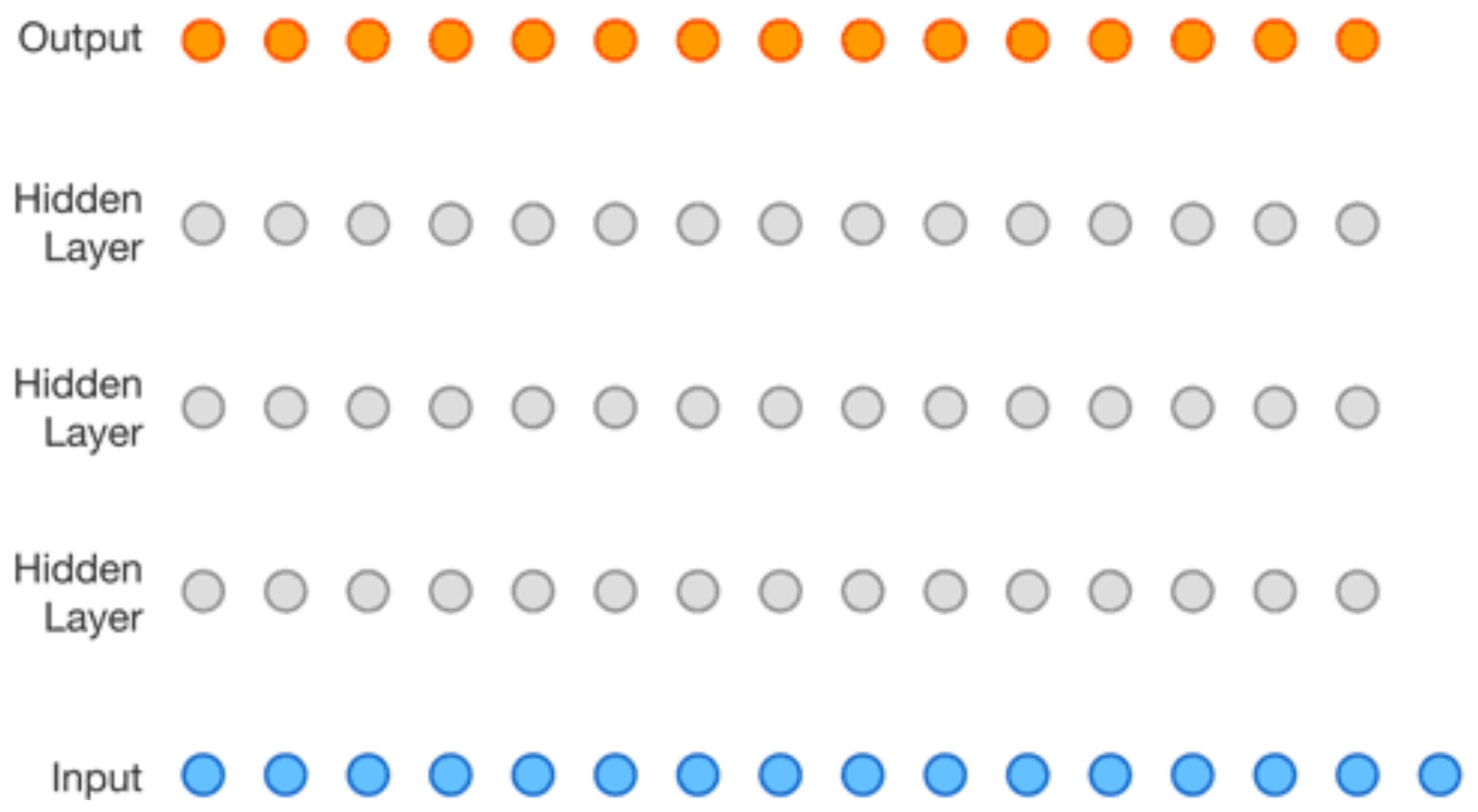


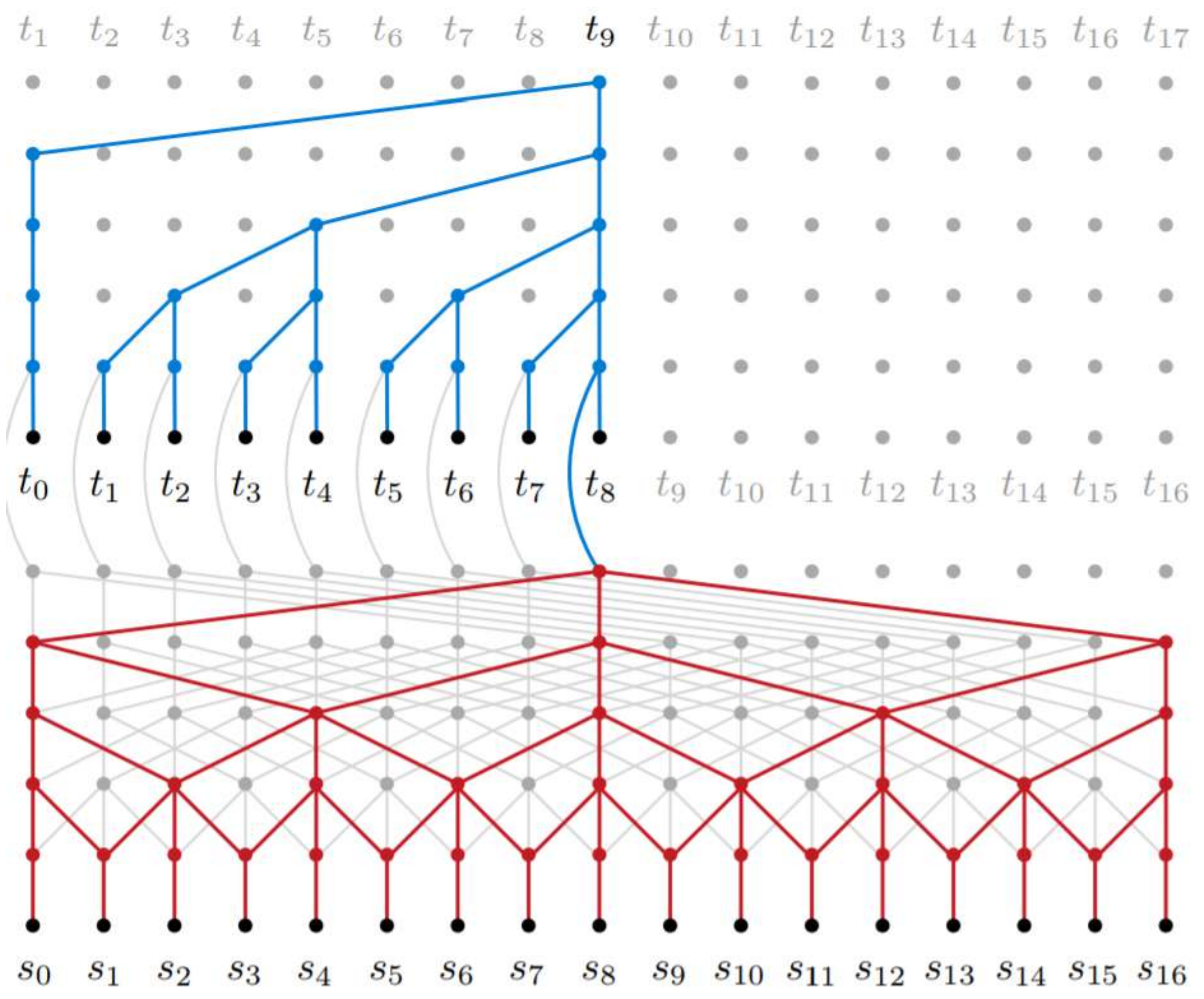
Image credit: https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/

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 непараллельной

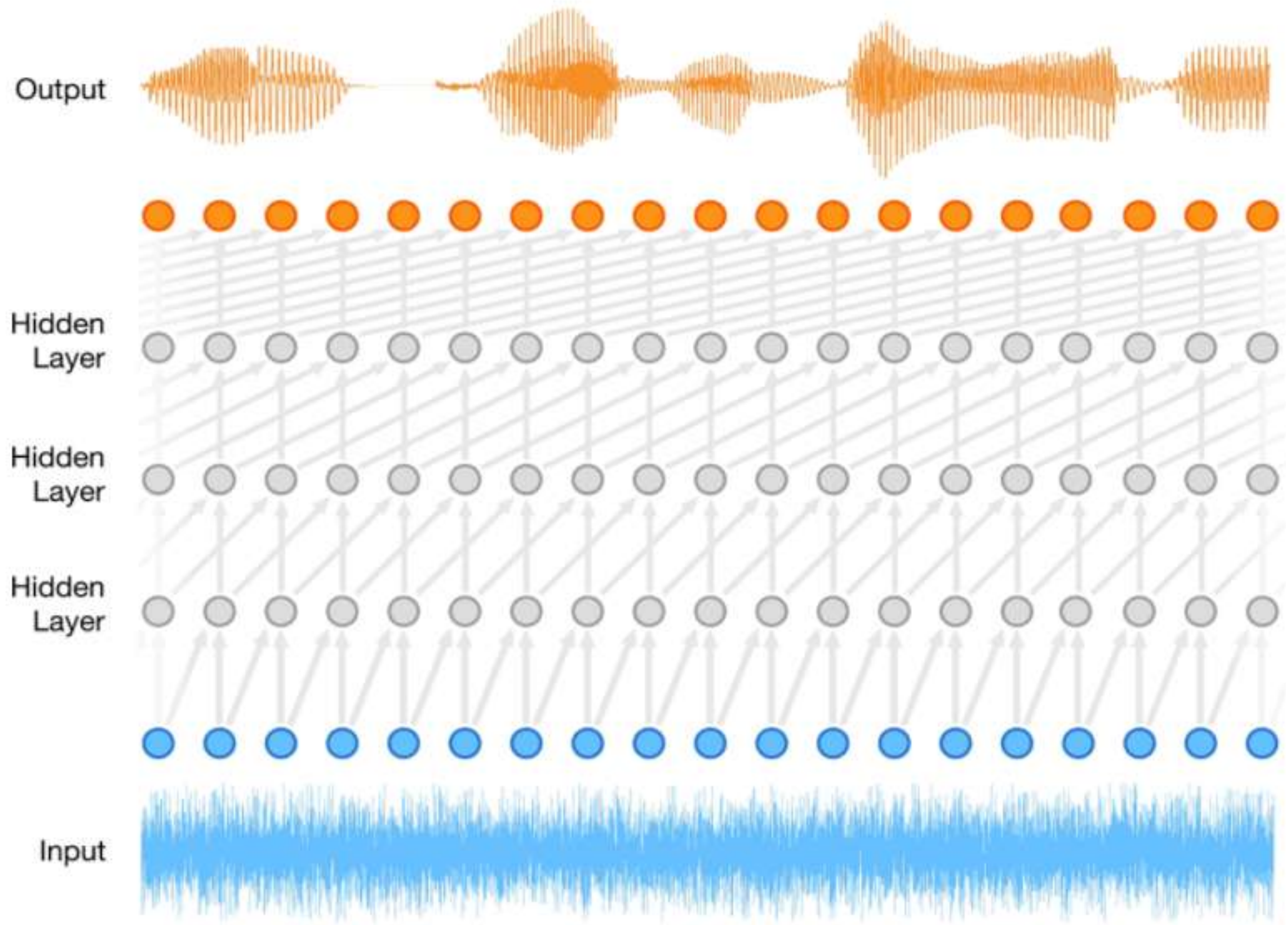


Image credit: https://deepmind.com/blog/high-fidelity-speech-synthesis-wavenet/

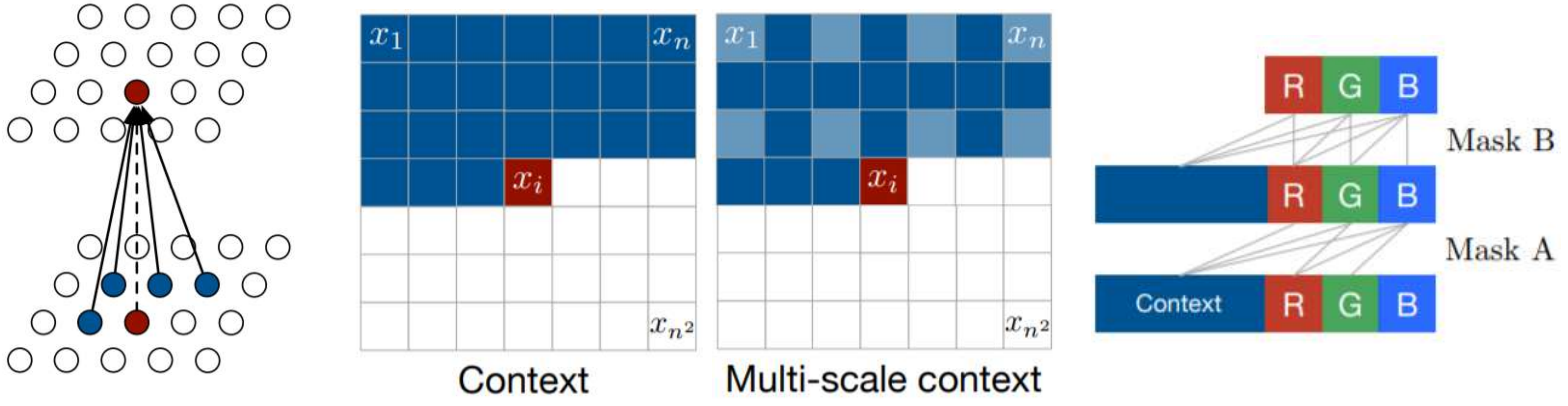
[van den Oord et al., 2017]

Как генерировать изображения? PixelCNN

- 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
- Упорядочить пиксели!
- 2D causal convolutions



- Нюансы архитектуры: residual, batchnorm, etc.

[van den Oord et al., 2016]

Generative Adversarial Networks

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

[Goodfellow et al., 2014]

Generative Adversarial Networks

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

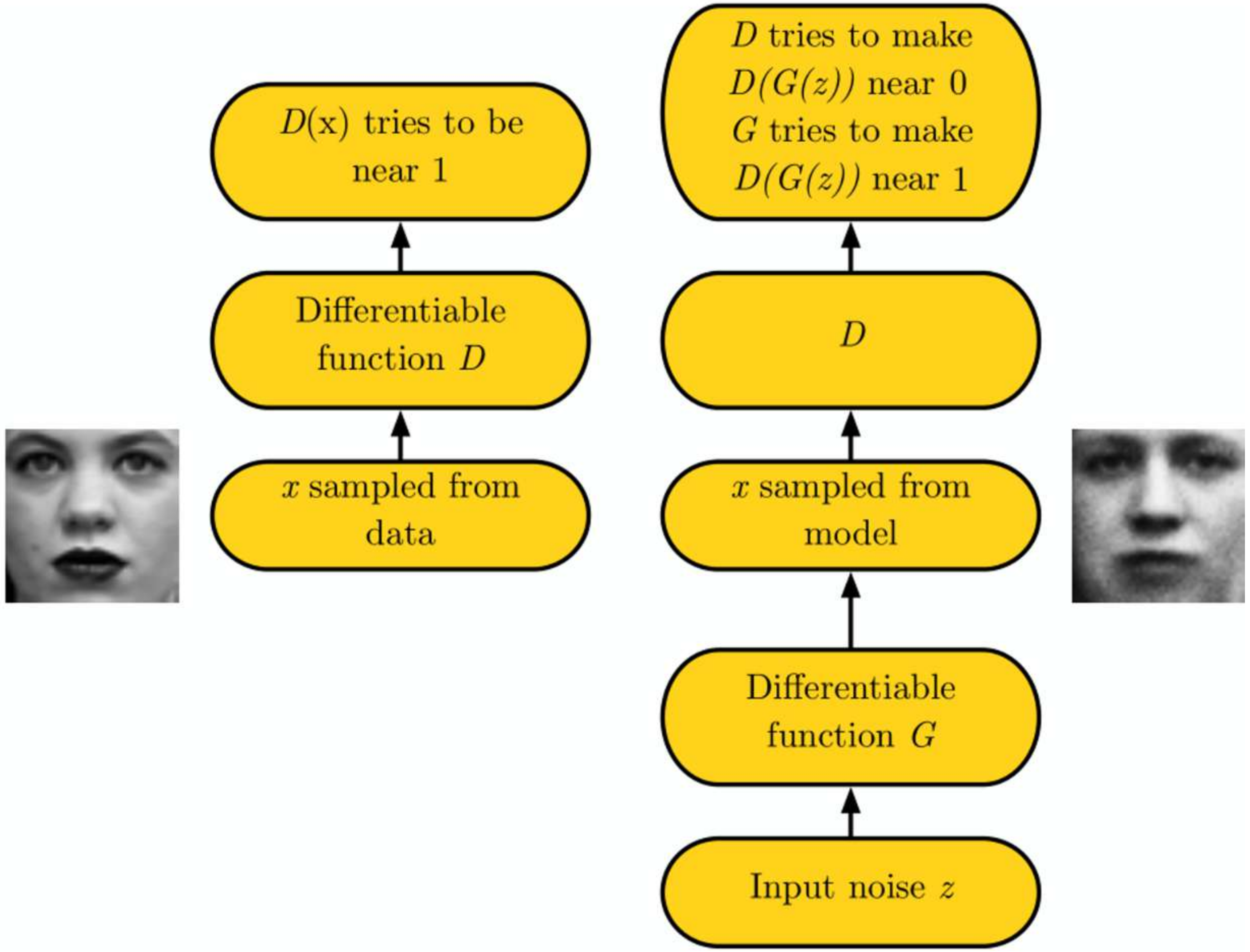


Image credit: Ian Goodfellow

[Goodfellow et al., 2014]

Generative Adversarial Networks

- 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?
  - f-дивергенции между распределениями

[Nowozin et al., 2016]

$$D_f(p_d \| p_m) = \int p_m(x) f\left(\frac{p_d(x)}{p_m(x)}\right) dx$$

$$\mathrm{KL} : f(u) = u \log(u)$$

  - Двойственное 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 некоторый класс T дают GAN

[Goodfellow et al., 2014]

Generative Adversarial Networks

- 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 шаге 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.

[Goodfellow et al., 2014]

Variational autoencoders (VAE)

- 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: $\max\limits_G \mathbb{E}_{z\sim q(z|x)} p(x, z|G)$
- Приближенный вывод!

Интеграл не берётся!

[Kingma&Welling, 2014]

Variational autoencoders (VAE)

[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!

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

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

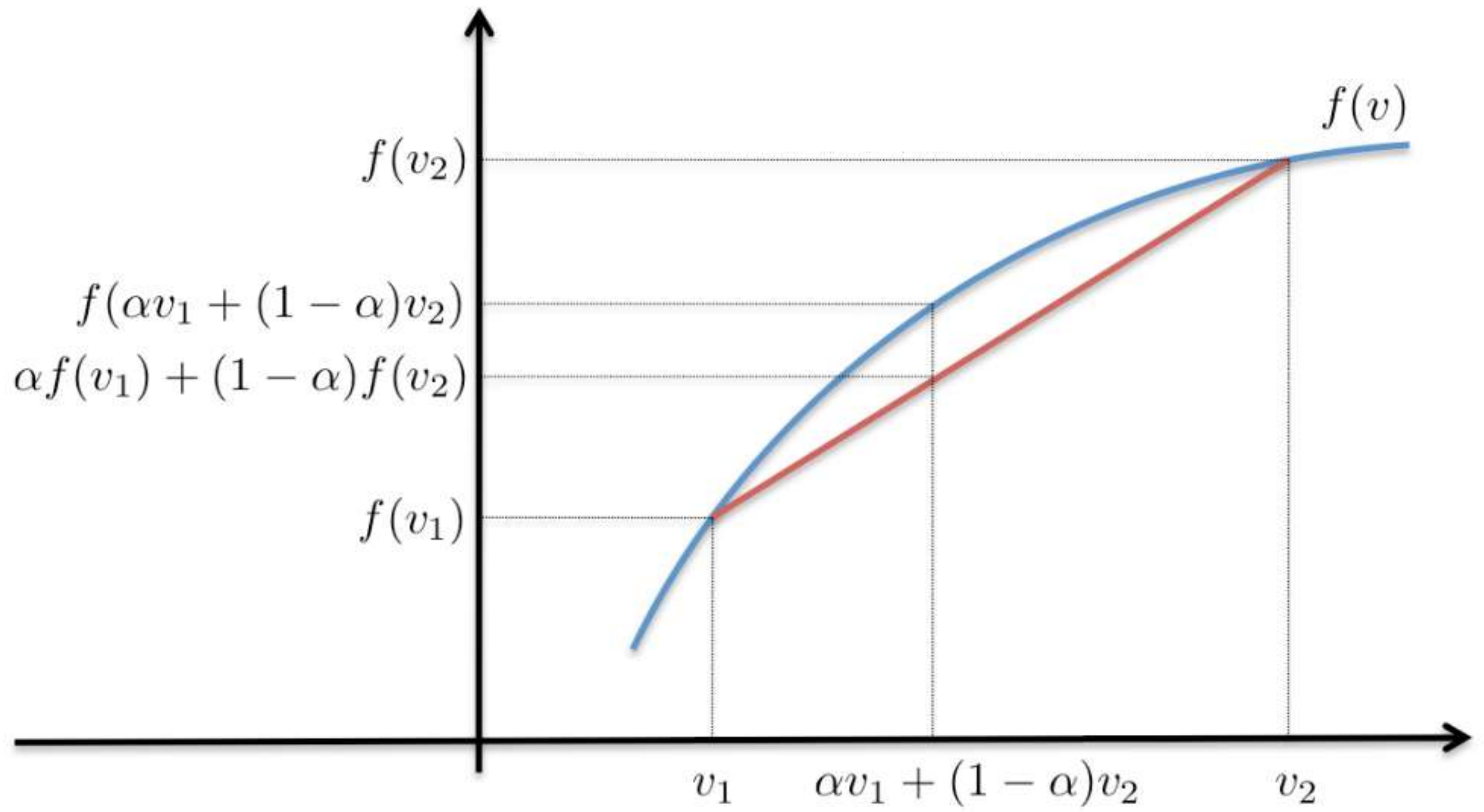
Jensen's inequality!

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

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

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

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



Variational autoencoders (VAE)

- 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!

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

- Будем 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!

[Kingma&Welling, 2014]

Градиент по распределению?

- Рассмотрим вероятностную модель

$z \sim p_z(z)$

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

- 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!

[Kingma&Welling, 2014]

Градиент по распределению?

- 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:

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

[Kingma&Welling, 2014]

Как работает VAE?

- Восстанавливает распределения!
- Но размытые картинки ☹

Image credit: Alec Radford

GANы [Karras et al. 2017]

Комбинации VAE и GAN

- Можно восстанавливать красивые изображения

  + получать мультимодальность

- Мультимодальный pix2pix

[Zhu et al., 2017]

Нормализационные потоки

- Можем ли мы напрямую оценивать правдоподобие?
    - Авторегрессионные модели сложны в обучении/семплировании
    - GAN не подходят
    - У VAE требуется проинтегрировать по z
- Ключевая идея: нужно обратимое преобразование f(z) из простого латентного распределения p(z)
- Тогда мы можем выразить плотность p(f(z)) как

$$p(x) = p(f^{-1}(x))\left|det\left(\frac{\partial f^{-1}(x)}{\partial x}\right)\right|$$

Якобиан отображения (должен быть обратимым)

Нормализационные потоки

- Обратимые преобразования можно комбинировать!
- Хорошие обратимые преобразования:
  - Нужен легко подсчитываемый определитель (треугольные матрицы etc.)
  - Пример: affine coupling

$$
\begin{aligned}
&\mathbf{x}_a, \mathbf{x}_b = \mathtt{split}(\mathbf{x}) \\
&(\log \mathbf{s}, \mathbf{t}) = \mathtt{NN}(\mathbf{x}_b) \\
&\mathbf{s} = \exp(\log \mathbf{s}) \\
&\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t} \\
&\mathbf{y}_b = \mathbf{x}_b \\
&\mathbf{y} = \mathtt{concat}(\mathbf{y}_a, \mathbf{y}_b)
\end{aligned}
$$

- Обучение: явно оптимизируем правдоподобие
- Генерация: x=f(z), z~p(z)
- Кодирование в латентное пространство: обращаем f

Заключение

- Авторегрессионные модели
  - Генерируют хорошие сэмплы (особенно текст и звук)
  - Нет скрытых представлений
- GAN
  - Могут генерировать красивые картинки
  - Проблемы с выучиванием распределений
- VAE
  - Хорошо восстанавливают распределения
  - Размытые картинки
- Нормализующие потоки
  - Можно считать правдоподобие в явной форме
  - Есть ограничения на архитектуры