НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

НАУЧНЫЙ ДОКЛАД

по результатам подготовленной
научно-квалификационной работы (диссертации)

ФИО Лобачева Екатерина Максимовна

Направление подготовки 09.06.01 Информатика и вычислительная техника

Профиль (направленность) программы 05.13.17 Теоретические основы информатики

Аспирантская школа по компьютерным наукам

Аспирант _____ / Лобачева Е.М. /
подпись

Научный руководитель _____ / Ветров Д.П. /
подпись

Директор Аспирантской школы по компьютерным наукам

_____ / Объедков С.А. /
подпись

Москва, 2018

Тема исследования: Применение методов глубинного обучения для поиска зависимостей в массивах структурированных данных

Актуальность темы исследования. Современные методы глубинного обучения показывают высокие результаты на многих задачах компьютерного зрения [1–5], обработки естественного языка [6–8], распознавания речи [9] и других. Наиболее эффективно глубинные нейронные сети показывают себя в том случае, когда архитектура этих сетей учитывает специфику структуры входных данных. Например, сверточные нейронные сети учитывают специфику связи пикселей в изображениях, а рекуррентные нейронные сети учитывают последовательную структуру данных в текстах.

В данной работе исследуются пути повышения качества работы и эффективности (по времени и по памяти) глубинных нейронных сетей для задач со структурированными данными. В частности, рассматриваются задачи с последовательными данными общего вида и с различными особенностями, а также задачи, в которых объекты состоят из нескольких компонент.

Многие практически важные задачи с последовательными данными (например, машинный перевод [7] и распознавание речи [9]) связаны с обработкой очень больших объемов данных, а также требуют сложных моделей для получения качественного решения. Повышение скорости работы таких моделей и их сжатие является важным и активно развивающимся направлением современных исследований [10–13]. В данной работе исследуются пути повышения эффективности рекуррентных нейронных сетей, являющихся одним из наиболее популярных типов нейросетевых архитектур для последовательных данных. Во-первых, рассматривается стратегия разреживания сетей для любого типа задач с последовательными данными. Во-вторых, рассматривается стратегия раннего предсказания для задачи классификации последовательностей, необходимая в задачах, связанных с распознаванием внешних событий для беспилотных автомобилей, мониторингом состояния пациентов, обнаружением аномалий в видеопотоке и других.

Также в данной работе исследуется применение нейросетевых архитектур к задаче классификации последовательностей с несимметричными классами в онлайн-режиме. Примером такой задачи может служить распознавание вредоносных программ по логу их выполнения прямо в процессе выполнения этих программ. Задача распознавания вредоносных файлов является очень важной для индустрии кибербезопастности. Несимметричность классов (по умолчанию объект относится к одному классу и только при наличии в нем определенных паттернов его можно относить к другому классу), последовательно-графовая структура данных (на лог можно смотреть и как на последовательность, и как на граф), а также отсутствие разметки для всех префиксов последовательностей (разметка есть только для полных последовательностей) накладывают определенные требования к методу классификации, которые нужно учитывать при подборе нейросетевой архитектуры для данного вида задач.

Кроме задач с данными последовательной структуры в работе также рассматриваются задачи, в которых объекты состоят из нескольких компонент. Примером такой задачи является генерация формы объектов на изображениях, так как объект обычно можно рассматривать как набор нескольких частей (лошадь состоит из головы, задних ног, передних ног и торса). Модели формы объекта играют важную роль в различных задачах компьютерного зрения: сегментации [14, 15], восстановлении части изображения [16], детектировании объектов [17].

Степень разработанности темы исследования. Специализированные нейросетевые архитектуры, учитывающие структуру данных, разрабатываются уже несколько десятков лет. Так, для работы с изображениями предложены сверточные нейронные сети [18], а для работы с последовательностями – рекуррентные нейронные сети [19]. Также на основе методов для работы с вероятностными графическими моделями предложены нейросетевые архитектуры для графовых данных [20]. В настоящее время подобные архитектуры активно развиваются и показывают высокие результаты на многих практически важных

задачах.

Применение современных нейронных сетей требует больших вычислительных затрат, поэтому вопросы их сжатия и ускорения являются важным направлением для исследований. Для ускорения и сжатия рекуррентных нейронных сетей применяются методы, основанные на специфичном представлении матриц весов [10, 11] или на применении эвристических методов постепенного обнулении весов с малыми по модулю значениями [12, 13]. Однако, для сетей прямого распространения и сверточных сетей также разработан теоретически обоснованный подход к разреживанию на основе обучения байесовской нейронной сети [21–23]. Данный подход показывает более высокие показатели по ускорению и сжатию нейронных сетей по сравнении с предыдущими, однако не применим напрямую к рекуррентным нейронным сетям. Задача адаптации данного подхода к рекуррентным нейросетям является открытой проблемой и рассматривается в данной работе.

Кроме упомянутых выше методов, для ускорения рекуррентных нейронных сетей применяются модели с адаптивным числом вычислений, которые могут производить вычисления не для всех элементов входных последовательности [24–26]. Большинство таких моделей не рассматривают возможности раннего предсказания (либо рассматривают, но никак не поощряют ранний останов), тогда как в некоторых задачах раннее предсказание не только ускоряет применение модели к данным, но и повышает качество модели и ее устойчивость при реальном применении в онлайн-режиме. Для поощрения раннего предсказания в рекуррентных нейросетях применяются методы, основанные на специальных регуляризациях [27, 28], однако они не позволяют определять момент предсказания автоматически. В данной работе рассматриваются пути добавления возможности автоматического определения момента предсказания и поощрения более ранних предсказаний в стандартные рекуррентные нейросети.

Задача распознавания вредоносных файлов по логам их выполнения является очень важной для индустрии кибербезопастности и поэтому активно иссле-

дуется [29–34], в том числе к ней применяются и методы глубинного обучения. Однако в онлайн-постановке, когда файлы нужно классифицировать по ходу их выполнения, эта задача ранее не рассматривалась. В данной работе исследуется возможность применения идеи монотонности в нейронных сетях для такой постановки, а также исследуются пути построения признакового описания логов с учетом их графовой структуры.

Модели формы объекта играют важную роль в различных задачах компьютерного зрения. Существует множество способов задавать форму объекта [15, 35, 36], однако большинство из них либо недостаточно гибкие, либо требуют сложной дополнительной разметки обучающей выборки. В [37] предлагается многоклассовая генеративная модель формы, позволяющая задавать форму объекта, состоящего из нескольких составляющих частей, с помощью глубинной сети определенной структуры. Данная модель обладает большой выразительной способностью, однако для ее обучения требуются данные с полной многоклассовой разметкой, которые сложны в получении. В данной работе рассматриваются пути уменьшения требований к разметке обучающей выборки для более эффективного обучения этой модели.

Цель и задачи исследования. Цель исследования: разработка методов глубинного обучения для более качественного и эффективного (по времени, памяти и объему необходимой разметки обучающих данных) решения задач со структурированными данными.

Для достижения поставленной цели были решены следующие задачи:

- Разработан метод разреживания рекуррентных нейросетевых архитектур на основе метода вариационного дропаута [21] для повышения эффективности (по времени и по памяти) данных архитектур для задач с последовательными данными.

- На основе метода адаптивного времени вычислений [38] разработан метод адаптивного раннего предсказания для повышения временной эффектив-

ности в задачах классификации последовательных данных.

- Разработан метод классификации последовательных данных для задач с несимметричными классами в онлайн-режиме, базирующийся на идее применения монотонной по последовательности модели. Подобный подход позволяет обучать модели для онлайн-постановки без дополнительной разметки префиксов последовательностей. Данный метод применен к задаче распознавания вредоносных программ по их логу выполнения, при этом также учтена графовая структура лога.

- Разработан метод обучения генеративной модели формы, который путем учета внутренней структуры формы объекта на изображении позволяет обучать более качественную модель формы по данным без дополнительной разметки.

Научная новизна. В данной работе впервые установлены следующие положения:

- Рекуррентные нейронные сети могут быть успешно разрежены с помощью метода вариационного дропаута [21] при условии модификации модели шума в нем с учетом последовательной структуры данных. Дополнительный стохастические переменные также позволяют эффективно и интерпретируемо сокращать входной словарь задачи и число нейронов и промежуточных гейтов в рекуррентных архитектурах.

- Момент раннего предсказания в задачах классификации последовательностей может автоматически выбираться моделью классификации адаптивно для разных входных последовательностей, что повышает качество и скорость работы рекуррентных сетей на подобных задачах.

- Монотонные нейронные сети показывают высокое качество в задаче классификации последовательных данных с несимметричными классами в он-

лайн-режиме, а также приводят к согласованным по времени и устойчивым предсказаниям.

- Многоклассовая больцмановская модель формы может быть обучена по данным с только бинарной разметкой путем применения аппарата вероятностных моделей с латентными переменными и вариационного EM-алгоритма.

Методология и методы исследования. В работе использованы методы машинного обучения, теории вероятностей, оптимизации, байесовских методов, компьютерной лингвистики и компьютерного зрения. Экспериментальное исследование проводится на языках Python и Cython, с использованием библиотек для глубинного обучения theano, pytorch и tensorflow, и удовлетворяет принципам воспроизводимости результатов.

1. Байесовское разреживание рекуррентных нейронных сетей

Большинство существующих методов разреживания рекуррентных нейронных сетей эвристичны и требуют аккуратного подбора большого числа гиперпараметров [12, 13] или аккуратного выбора структуры сжатия сети [10, 11]. В данной главе рассматривается адаптация теоретически обоснованного метода разреживания нейросетей, вариационного дропаута (SparseVD) [21], на случай рекуррентных нейронных сетей с учетом специфики их структуры, а также модификация полученной модели для дополнительного разреживания входного словаря (который во многих задачах приводит к наибольшим затратам по памяти), нейронов и промежуточных гейтов в рекуррентных архитектурах.

Нотация. Обозначим входную последовательность за $x = [x_0, \dots, x_T]$, правильное и выдаваемое рекуррентной сетью предсказания для этой последова-

тельности за y и $\hat{y}$ соответственно (они могут быть как скалярами в случае задачи классификации последовательностей, так и векторами в случае задачи генерации последовательности). За X, Y обозначим обучающую выборку $\{(x^1, y^1), \ldots, (x^N, y^N)\}$. Все веса рекуррентной нейронной сети, кроме векторов сдвига, обозначим за ω, тогда как один вес из этого набора – за w_{ij}. Вектора сдвига обозначаются отдельной переменной B, так как они не разреживаются в предлагаемой модели.

Для примера, простая архитектура для задачи генерации последовательности выглядит следующим образом:

$$\begin{aligned}
&\text{слой представлений} : \tilde{x}_t = w^e_{x_t},\\
&\text{рекуррентный слой} : h_{t+1} = \sigma(W^h h_t + W^x \tilde{x}_{t+1} + b^r),\\
&\text{полносвязный слой} : \hat{y}_t = \text{softmax}(W^d h_t + b^d),
\end{aligned}$$

где $y = [x_1, \ldots, x_T]$. Здесь $\omega = \{W^e, W^x, W^h, W^d\}$, $B = \{b^r, b^d\}$.

Общий метод разреживания. Для применения байесовских техник к разреживанию нейронных сетей, эта сети рассматриваются как байесовские нейронные сети, в которых веса не оцениваются точечно, а описываются апостериорным распределением. Следуя [21, 39], на веса модели накладывается полностью факторизованное априорное распределение:

$$p(\omega) = \prod_{w_{ij} \in \omega} p(w_{ij}), \quad p(w_{ij}) \propto \frac{1}{|w_{ij}|} \tag{1}$$

и апостериорное распределение приближается в полностью факторизованном семействе вариационных распределений:

$$q(w|\theta, \sigma) = \prod_{w_{ij} \in \omega} \mathcal{N}\big(w_{ij}|\theta_{ij}, \sigma^2_{ij}\big). \tag{2}$$

Задача получения приближения на апостериорное распределение

$$\min_{\theta, \sigma, B} KL(q(\omega|\theta, \sigma)||p(\omega|X, Y, B)) \tag{3}$$

эквивалентна оптимизации вариационной нижней оценки [21]:

$$-\sum_{i=1}^{N}\int q(\omega|\theta,\sigma)\log p(y^i|x_0^i,\ldots,x_T^i,\omega,B)d\omega+$$
$$+\sum_{w_{ij}\in\omega} KL(q(w_{ij}|\theta_{ij},\sigma_{ij})||p(w_{ij}))\to\min_{\theta,\sigma,B}. \quad (4)$$

Здесь первый член приближается с помощью одного сэмпла из $q(\omega|\theta,\sigma)$. Второй член является регуляризатором, поощряющим разреженность весов, и может быть с большой точностью приближен аналитически функцией, зависящей от θ и σ [21].

Для получения несмещенной оценки на градиент оптимизируемого функционала используется трюк репараметризации [40]:

$$w_{ij}=\theta_{ij}+\sigma_{ij}\epsilon_{ij},\quad \epsilon_{ij}\sim\mathcal{N}(\epsilon_{ij}|0,1) \quad (5)$$

Для корректного применения описанной модели к рекуррентным нейронным сетям важно использовать один и тот же сэмпл весов для всех моментов времени t при подсчете правдоподобия $p(y^i|x_0^i,\ldots,x_T^i,\omega,B)$ [41, 42]. Эта особенность связана с тем, что в рекуррентных нейронных используется один и тот же набор весов для всех моментов времени.

Также в случае рекуррентных нейронных сетей не везде применима локальная репараметризация [21, 39], состоящая в том, что вместо весов генерируются преактивации нейронов, например:

$$(W^x x_t)_i=\sum_j\theta_{ij}^x x_{tj}+\epsilon_i\sum_j(\sigma_{ij}^x)^2x_{tj}^2. \quad (6)$$

Локальная репараметризация не применима к матрицам весов, которые используются больше чем в одном моменте времени, из-за ограничения на связанную по моментам времени генерацию таких матриц.

Для матрицы W^h линейная комбинация $(W^h h_t)$ не распределена нормально, так как h_t зависит от W^h с предыдущего шага. В результате, правило о

сумме независимых нормально распределенных величин с константными коэффициентами не применимо. На практике рекуррентные сети с локальной репараметризацией для $(W^h h_t)$ не могут быть качественно обучены.

Для матрицы W^x линейная комбинация $(W^x x_t)$ распределена нормально, однако генерация одинаковой матрицы W^x для всех моментов времени и генерация одинакового шума ϵ_i для всех моментов времени на эквивалентно. На практике рекуррентные сети с локальной репараметризацией для $(W^x x_t)$ могут работать так же хорошо по качеству, как и без локальной репараметризации, и их обучение даже сходится быстрее в некоторых случаях.

В результате, обучение выглядит следующим образом. На проходе по сети вперед генерируются веса ω по формуле (5) и далее рекуррентная сеть применяется стандартным образом. На проходе назад подсчитываются градиенты функционала ошибки по $\theta, \log\sigma, B$, которые после используются для оптимизации.

На этапе тестирования используется детерминированная сеть со средним значением θ в качестве весов. При этом регуляризатор (второй член в (4)) приводит к тому, что многие элементы θ сходятся к низким по модулю значениям. За счет этого сеть может быть разрежена путем обнуления тех весов, для которых отношение сигнала к шуму ниже некоторого порога: $\frac{\theta_{ij}^2}{\sigma_{ij}^2} < \tau$(так же делается в [21]).

Разреживание входного словаря. Одно из преимуществ байесовских методов разреживания состоит в том, что они могут быть обобщены на случай групповой разреженности любых групп весов без значительного усложнения процедуры обучения за счет введения новых стохастических мультипликативных весов для таких групп [23]. В данной главе предлагается дополнительно разреживать входной словарь модели и, как следствие, матрицу весов входного слоя, так как эта матрица во многих задачах с последовательностями является наибольшей частью модели по занимаемой памяти.

Для разреживания входного словаря предлагается ввести дополнительные стохастические веса $z \in \mathbb{R}^V$ для слов в словаре (здесь за V обозначен размер словаря). Проход вперед по сети при этом выглядит следующим образом:

1. генерируется вектор z^i из текущего приближения апостериорного распределения для каждой входной последовательности x^i в мини-батче;
2. каждый токен x_t^i из последовательности x^i, закодированный бинарным вектором, умножается на z^i;
3. продолжается проход вперед в стандартном режиме.

В остальном работа с весами z производится так же, как и с остальными весами W: используется лог-равномерное априорное распределение, а апостериорное распределение приближается в семействе полностью факторизованных нормальных распределений с обучаемыми средними и дисперсиями. После обучения элементы z с низким отношением сигнала к шуму могут быть обнулены, а соответствующие им элементы словаря и столбцы матрицы входного слоя убраны из модели.

Разреживание нейронов и гейтов. Для ускорения нейронных сетей выгодно не просто разреживать матрицы весов, а делать это в групповой манере, когда из модели убираются нейроны целиком. В данной главе рассматривается техника разреживания сетей как на уровне нейронов, так и на уровне промежуточных гейтов в рекуррентных архитектурах с гейтами, таких как LSTM [43].

LSTM представляет собой расширенную версию стандартной рекуррентной архитектуры. Подсчет выхода ячейки LSTM в момент времени t производится по следующим формулам:

$$i = \sigma(W_i^h h_{t-1} + W_i^x x_t + b_i) \quad f = \sigma(W_f^h h_{t-1} + W_f^x x_t + b_f)$$
$$g = tanh(W_g^h h_{t-1} + W_g^x x_t + b_g) \quad o = \sigma(W_o^h h_{t-1} + W_o^x x_t + b_o)$$
$$c_t = f \odot c_{t-1} + i \odot g \qquad h_t = o \odot tanh(c_t)$$

Здесь вектор x_t является входным вектором, вектора h являются скрытыми векторами (и выходными векторами рекуррентного слоя), матрицы W и вектора b являются обучаемыми параметрами модели, переменные i, o, f соответствуют промежуточным гейтам (входному, выходному и гейту забывания), переменной g обозначается информационный поток, а переменной c – вектор памяти.

По аналогии с техникой разреживания входного словаря предлагается ввести дополнительные мультипликативные стохастические переменные z^x и z^h на вектора x_t и h_{t-1} для разреживания входных и скрытых нейронов, а также переменные z^i, z^f, z^g и z^o на преактивации промежуточных гейтов и информационного потока для разреживания этих гейтов и информационного потока. При применении этих переменных подсчет выхода ячейки LSTM изменяется следующим образом:

$$
\begin{aligned}
i = \ & \sigma\Big(\big(\hat{W}_i^h(h_{t-1} \odot z^h) + \hat{W}_i^x(x_t \odot z^x)\big) \odot z^i + b_i \Big) \\
f = \ & \sigma\Big(\big(\hat{W}_f^h(h_{t-1} \odot z^h) + \hat{W}_f^x(x_t \odot z^x)\big) \odot z^f + b_f \Big) \\
g = \ & tanh\Big(\big(\hat{W}_g^h(h_{t-1} \odot z^h\big) + \hat{W}_g^x(x_t \odot z^x)\big) \odot z^g + b_g \Big) \\
o = \ & \sigma\Big(\big(\hat{W}_o^h(h_{t-1} \odot z^h) + \hat{W}_o^x(x_t \odot z^x)\big) \odot z^o + b_o \Big) \\
c_t \ & = f \odot c_{t-1} + i \odot g \qquad h_t = o \odot tanh(c_t)
\end{aligned}
$$

Обнуление компонент z^x и z^h приводит к тому, что из модели убираются соответствующие элементы входного или скрытого вектора, а также соответствующие строки в матрицах весов. Обнуление компонент z^i, z^f, z^g или z^o приводит к тому, что элементы соответствующего гейта или информационного потока становятся константным, не обусловленными на входные данные x_t и h_t. Появление в модели константных гейтов упрощает, но не нарушает структуру LSTM, и кроме того, позволяет экономить вычисления матричных произведений.

Работа с добавленными переменными z производится так же, как и с остальными весами W и переменными z для разреживания словаря.

Таблица 1: Результаты для посимвольной генерации текста. Сравнение предлагаемого метода SparseVD, обученного из различных начальных приближений, с базовой архитектурой и регуляризованной архитектурой (VDB).

Метод (иниц.)	Валид.	Тест	Разреженность W^x - W^h - W^y %
Базовый	1.499	1.453	–
VBD для W^h	**1.394**	**1.358**	–
SparseVD (случайная)	1.506	1.461	**79.7 – 88.0 – 71.7**
SparseVD (без дропаута)	1.454	1.414	61.9 – 60.0 – 43.9
SparseVD (VBD для W^h)	1.409	1.372	52.2 – 49.8 – 37.9

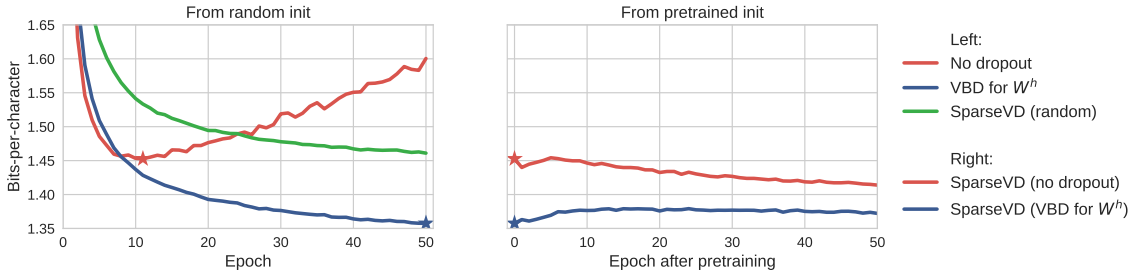


Рис. 1: Результаты для посимвольной генерации текста. Слева – обучение из случайной инициализации, справа – из предобученных моделей.

Эксперименты. Эксперименты проводились на задаче генерации и классификации текстов. В данной главе приводятся в основном результаты на первом типе задач. Рассматривается посимвольная и пословная генерация текста, в качестве данных выбран корпус Penn Treebank [44] со стандартным разбиением на обучение, валидацию и контроль [45].

Во-первых, в экспериментах было показано, что предложенная техника приводит к высокому уровню разреживания без сильной потери в качестве решения задачи (табл. 1), а также хорошо регуляризует модель и помогает бороться с проблемой переобучения (рис. 1). Также предварительное обучение модели без разреживания позволяет значительно сократить потери в качестве

Таблица 2: Результаты для посимвольной и пословной генерации текста. Уровень сжатия: $|w|/|w \neq 0|$. SparseVD – разреживаются только веса, SparseVD-Voc – дополнительно разреживается словарь. Для посимвольной задачи качество приводится в битах на символ, для пословной – в величине перплексии.

Задача	Метод	Валид.	Тест	Сжатие	Словарь	Нейроны h
Char PTB	Original	1.498	1.454	1x	50	1000
	SparseVD	1.472	1.429	**4.2x**	50	**431**
	SparseVD-Voc	**1.458**	**1.417**	3.53x	**48**	510
Word PTB	Original	135.6	129.5	1x	10000	256
	SparseVD	**115.0**	**109.0**	**14.0x**	9985	**153**
	SparseVD-Voc	126.3	120.6	11.1x	**4353**	207

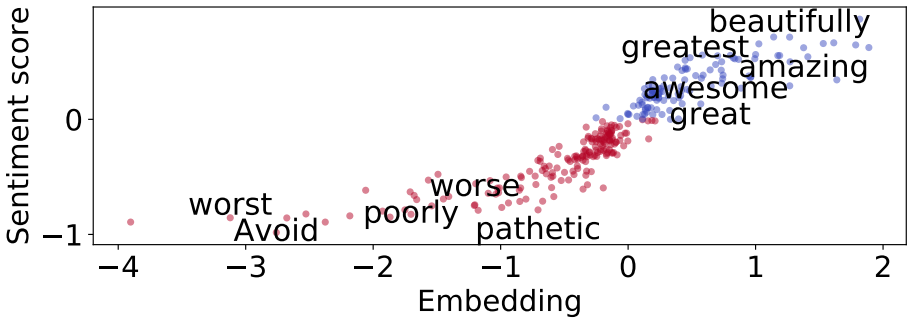


Рис. 2: Данные IMDB: оставшиеся элементы словаря, по горизонтальной оси отложен соответствующий вес из входного слоя модели, по вертикальной – тональность слов ((#полож. - #отриц.) / #общее число текстов с этим словом).

при последующем разреживании.

Во-вторых, показано, что дополнительное разреживание словаря приводит к значительному увеличению общего уровня разреженности модели (таб. 2). Также разреживание словаря приводит к интерпретируемым результатам. Например, в случае задачи классификации текстов по их эмоциональной окраске, в модели остаются только основные положительные и отрицательные слова (хороший, исключительный и т.п.), а соответствующие им веса во входном слое сильно коррелируют с их тональностью (рис. 2).

Таблица 3: Сравнение SparseVD и SparseVD с дополнительным разреживанием нейронов и гейтов (SparseVD-group) с другими методами группового разреживания на задаче классификации текстов IMDB.

Модель	Точность	Нейроны x	Нейроны h	Гейты
Базовая	0.841	300	128	512
Lognormal noise [22]	**0.846**	6	77	308
Bayes. Compression [23]	0.833	4	17	68
Sparse Structures [13]	0.816	9	4	16
SparseVD	0.82	3	8	17
SparseVD-group	0.823	**1**	**4**	**8**

В-третьих, показано, что дополнительное разреживание нейронов и гейтов приводит к лучшим результатам по соотношению качества и уровня разреженности, по сравнению с другими существующими методами группового разреживания (табл. 3).

2. Адаптивное время предсказания

Задача раннего предсказания в классификации последовательностей заключается в том, что объект-последовательность подается классификатору постепенно и важно не только выдать правильную метку для данного объекта, но и сделать это как можно раньше. Сложность данной задачи состоит в необходимости соблюдать баланс между качеством классификации и скоростью ее получения.

Рекуррентные нейронные сети являются стандартной нейросетевой архитектурой для задач с последовательными данными. Однако в них не заложен функционал определения момента времени, в который пора делать предсказание, и поощрения более ранних предсказаний. В данной главе предлагается

метод адаптивного времени предсказаний для рекуррентных нейросетей, основанный на методе адаптивного времени вычислений [38].

Метод. Предлагаемый метод адаптивного времени предсказаний (APT) заключается в добавлении дополнительного остановочного блока в сеть, который запускается на каждом шаге при прочтении последовательности и отвечает за вероятность остановки и выдачи предсказания после текущего шага. Остановочный блок представляет собой нейрон с сидмоидальной функцией активации, на вход которому подается скрытое состояние рекуррентного слоя. Данный остановочный блок вдохновлен идеей из метода адаптивного времени вычислений [38] и может быть добавлен в любую рекуррентную нейронную сеть. Для примера рассмотрим нейронную сеть, состоящую из одного рекуррентного слоя и одного выходного softmax слоя.

Обозначим за x_t, s_t входной и скрытый вектора сети в момент времени t, а за θ – вектор параметров сети. На каждом шаге t при чтении входной последовательности модель сначала пересчитывает скрытое состояние рекуррентного слоя, после чего подает его в остановочный блок для подсчета *оценки остановки* h_t:

$$s_t = RNN(x_t, s_{t-1}, \theta), \qquad h_t = \sigma(W_h s_t + b_h), \tag{7}$$

где W_h, b_h являются обучаемыми параметрами.

Результирующая *оценка остановки* h_t далее используется для подсчета весов p_t для каждого элемента входной последовательности, которые называются *вероятностями остановки*:

$$p_t = \begin{cases} R, & t = N \\ h_t, & t < N \end{cases}, \text{ где} \tag{8}$$

$$N = \min\left\{t' : \sum_{t=1}^{t'} h_t \geqslant 1 - \varepsilon,\ T\right\}, \quad R = 1 - \sum_{t=1}^{N-1} h_t \tag{9}$$

Модель останавливается после шага N, когда сумма оценок остановки по пройденным моментам времени превышает $1 - \varepsilon$, где ε – маленькая константа, позволяющая остановить вычисления даже после первого шага. Таким образом, вектор из $\{p_t | t = 1, \ldots, N\}$ представляет собой вектор вероятностей остановки после конкретного шага в последовательности. Для суммирования данного вектора в единицу вводится остаток R.

После N шагов модель классифицирует последовательность используя softmax слой, на вход которому подается взвешенный вектор скрытых состояний:

$$\hat{s} = \sum_{t=1}^{N} p_t s_t \tag{10}$$

Таким образом вероятности p_t отвечают не только за момент остановки, но и за важность определенных частей последовательности для предсказания, по аналогии с механизмом внимания для нейронных сетей [46].

Для обучения описанной модели минимизируется стандартный функционал ошибки для задачи классификации с добавлением специального регуляризатора, отвечающего за поощрение раннего предсказания. Для одного объекта функция потерь выглядит следующим образом:

$$\hat{L}(y, \hat{y}) = L(y, \hat{y}) + \lambda R. \tag{11}$$

Так как N является кусочно постоянной функцией от оценок остановки и сложна в оптимизации, в качестве регуляризатора берется остаток R. Так как R является линейной функцией от оценок остановки для всех моментов времени, кроме последнего, градиент функции потерь выглядит следующим образом:

$$\frac{\partial \hat{L}}{\partial h_t} = \frac{\partial L}{\partial h_t} + \lambda \cdot \begin{cases} 0, & t = N, \\ -1, & t < N \end{cases} \tag{12}$$

Таким образом, оптимизация предложенной функции потерь приводит к увеличению оценки остановки для всех моментов времени кроме последнего и поэтому поощряет ранний останов.

Таблица 4: Результаты раннего предсказания.

Данные	**Модель**	**Точность**	**Число шагов**	**Среднее N**
SeqMNIST	GRU	97.4	784	784
	Skip GRU	97.5	**393.78**	783.18
	REINFORCE	98.64	678	678
	APT, $\lambda = 10^{-2}$	98.55	542.9	**542.9**
	APT, $\lambda = 0$	**98.85**	635.1	635.1
Reordered SeqMNIST	GRU	11.32	784	784
	Skip GRU	89.45	282.28	774.12
	REINFORCE	87.5	717.9	717.9
	APT, $\lambda = 10^{-2}$	**92.3**	**262.4**	**262.4**
	APT, $\lambda = 0$	91.47	324.4	324.4
UCF-101	GRU	**81.7**	250	250
	Skip GRU	79.2	**29.7**	-
	APT, $\lambda = 10^{-5}$	79.3	85.43	**85.43**
	APT, $\lambda = 10^{-6}$	79.59	90.72	90.72

Эксперименты. Эксперименты проводились на нескольких наборах данных: SeqMNIST – последовательный MNIST [47] (изображения цифр, рассматриваемые как последовательности пикселей), его вариант с отсортированными по изменчивости пикселями (Reordered SeqMNIST) и набор коротких видео UCF101 [48]. В первых двух случаях решалась задача классификации цифр, в последнем – распознавание действий на видео.

Во-первых, в экспериментах было показано, что предложенная техника раннего предсказания превосходит аналоги [24, 26, 49] по соотношению качества и времени предсказания (табл. 4). Однако данная техника показывает высокие результаты только в задачах, в которых выполняется предположение о возможности раннего предсказания. Поэтому результаты предложенного метода на Reordered SeqMNIST значительно превышают аналогичные на обычном

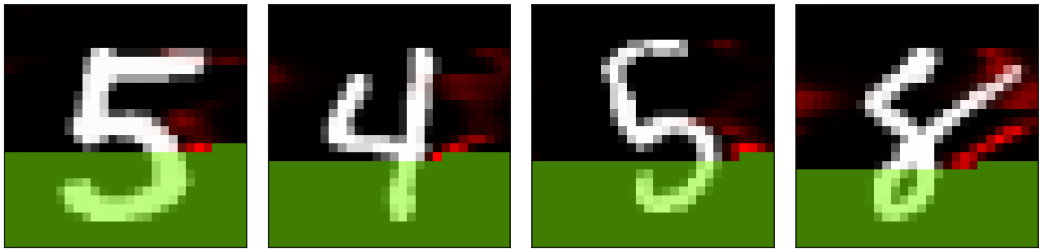


Рис. 3: Вероятности остановки для SeqMNIST. Красный соответствует высоким p_t. Пиксели, не используемые моделью, отмечены зеленым.

SeqMNIST.

Во-вторых, показано, что вероятности остановки ведут себя интерпретируемо и, например, в задаче классификации цифр большие значения вероятностей соответствуют ключевым точкам и границам объектов на изображениях (рис. 3).

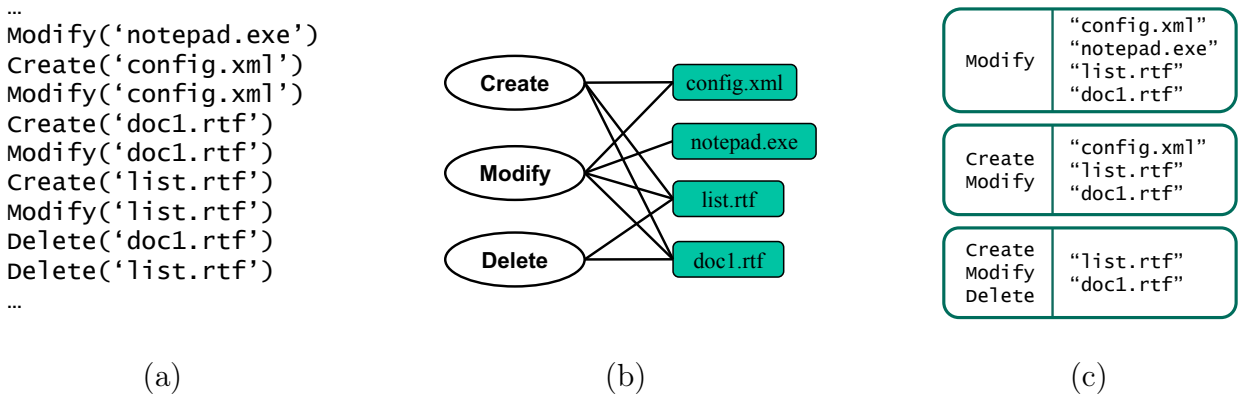
3. Монотонные модели для распознавания вредоносных файлов

Распознавание вредоносных файлов по логу их выполнения является основным методом динамического анализа вредоносного кода. Лог обычно представляет из себя последовательность наблюдаемых системных событий, описываемых названием события, его аргументами и, возможно, возвращаемыми значениями. Большинство существующих методов основано на извлечении простого набора признаков, таких как n-граммы событий или бинарные индикаторы связи между событиями и их аргументами, и дальнейшего применения стандартных методов классификации [29–32]. Также существуют методы, которые учитывают последовательную структуру логов и применяют рекуррентные нейронные сети [33, 50]. Все приведенные методы решения данной задачи не учитывают графовую структуру лога, что приводит к потере важной структурной информации.

Также упомянутые методы рассматривают только постановку, в которой

классификатор принимает на вход весь лог единовременно, поэтому их применение в онлайн режиме, когда лог подается классификатору постепенно, приводит к неконсистентным по времени результатам: в один момент времени классификатор может быть уверен в том, что программа вредоносная, а уже в следующий считать, что она безвредная. Расширение обучающей выборки путем добавления префиксов логов в качестве объектов не может решить описанную проблему, так как метки для префиксов вредоносных логов неизвестны (вредоносная программа может вести себя абсолютно безвредно на протяжении очень долгого периода времени в начале исполнения). Более того, существующие методы могут использовать так называемые «обеляющие» признаки: это такие события в логе, видя которые модель начинает выдавать предсказание, более близкое к чистому файлу. Использование таких признаков делает модель уязвимой к атакам на систему. Например, если запуск какого-то процесса считается моделью «обеляющим» признаком, то для того, чтобы обмануть модель, достаточно добавить несколько запусков этого процесса во вредоносную программу.

В данной главе предлагается новая техника построения признакового описания лога, основанная на построении поведенческого графа и выделении патернов из этого графа. Также предлагается модифицировать модель распознавания таким образом, чтобы выделение признаков и классификация были монотонны в том смысле, что добавление любых новых строчек в лог могло бы приводить только к увеличению вероятности того, что модель отнесет данный лог к вредоносным. Такое ограничение приводит к тому, что предсказанная моделью вероятность вредоносности может только увеличиваться с ходом выполнения программы. То есть, для чистого файла предсказанная вероятность вредоносности должна быть низкой в течение всего времени выполнения программы, а для вредоносного файла она должна увеличиться при наблюдении подозрительной активности и оставаться на высоком уровне до конца лога. Дополнительно, ограничение монотонности приводит к невозможности использования моделью «обеляющих» признаков и делает предсказания модели устойчивыми к добав-



(a) (b) (c)

Рис. 4: (a) – пример лога, (b) – поведенческий граф, (c) – паттерны.

лению любой новой активности в лог.

Признаковое описание логов. В данной главе лог представляет собой последовательность системных событий со своими аргументами (простой пример на рис. 4a). Так как представление лога в виде последовательности не устойчиво к мультипроцессингу, обфускации кода и разнице окружения запуска программы, предлагается представлять лог в виде поведенческого графа (рис. 4b). В таком графе каждое событие и аргумент представляются в виде вершин, а связи между вершинами задаются исходя из того, какие события и аргументы встречаются вместе в логе.

Для построения признакового описания поведенческого графа из него сначала извлекаются поведенческие паттерны (рис. 4c). Паттерн представляют собой группу событий и аргументов, которая отражает определенный тип поведения программы. Паттерны могут быть построены как монотонным образом (на рис. 4c именно такие), так и немонотонным образом. Для построения немонотонных паттернов сначала для каждого аргумента отбираются все связанные с ним события, после чего аргументы с одинаковым набором событий объединяются в паттерн (события при этом тоже входят в него). Для построения монотонных паттернов эта процедура изменяется таким образом, чтобы выполнялось следующее правило: для любого аргумента не существует такого паттерна, что все входящие в него события связаны с данным аргументом, но при этом сам

аргумент в него не входит.

Далее, для каждого паттерна строится его численное признаковое описание. Для этого сначала паттерн представляется в виде бинарного вектора, часть которого отвечает за наличие в нем определенных событий, а другая часть – определенных частей аргументов (аргументы разделяются на отдельные токены по разделяющим символам, таким как ’://’,’.’,’:’). При этом события и токены берутся из конечного словаря. После этого бинарный вектор подается на вход автоэнкодеру в случае, когда мы хотим обучать признаки без учителя, или однослойной нейронной сети в случае, когда выделение признаков является только частью задачи классификации и выполняется вместе с ней. В результате для каждого паттерна получается плотный вектор признаков низкой размерности.

Для получения признакового описания поведенческого графа фиксированного размера из признаков составляющих его паттернов применяется слой динамического пулинга в нейронной сети: по каждой размерности вектора признаков для паттернов применяется одна из операций min, max, mean (или все они) относительно разных паттернов.

Описанная схема выделения признаков для логов может применяться как в режиме без учителя (в этом случае функционал качества при обучении строится на основе качества реконструкции в автоэнкодере для паттернов), так и в режиме обучения с учителем как первый шаг для решения задачи классификации. Если во втором случае в качестве алгоритма классификации выбрать нейронную сеть, то всю процедуру выделения признаков, начиная с шага выделения признакового описания для паттернов, и процедуру классификации можно представить в виде одной большой нейросетевой архитектуры и обучать совместно.

Эксперименты. Эксперименты проводились на большом наборе реальных логов, состоящем из примерно 8 млн. файлов, разбитых в соотношении 7:3 на

ТОКЕН	ТОП-5 БЛИЖАЙШИХ СОСЕДЕЙ
word	excel, dotm, outlook, machine, open
com	www, ://, http, net, ru
jpg	png, gif, css, xml, html
js	txt, htm, ie5, css, cookies
34	36, 42, 56, 38, 35
3960	3964, 3972, 3952, 3968, 3956

Таблица 5: Структура признакового пространства.

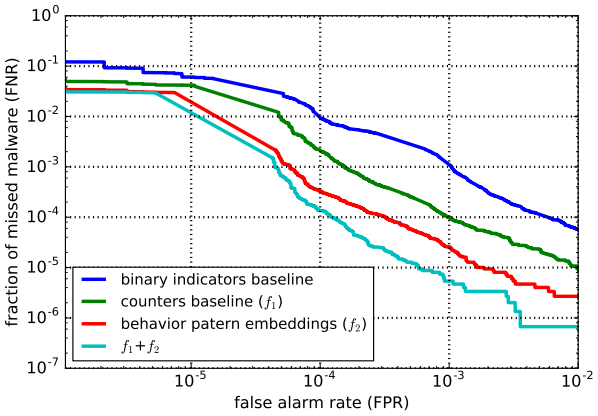


Рис. 5: Сравнение признаковых описаний логов (f_2 – предлагаемый метод).

обучение и контроль.

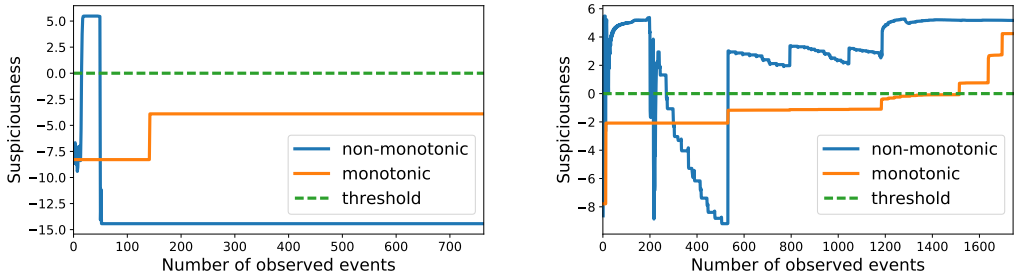
Во-первых, в экспериментах было показано, что предложенная техника построения признакового описания для логов приводит к интерпретируемой структуре в пространстве признаков для токенов (признаки для токенов могут быть извлечены как строки матрицы весов из слоя нейросети, который по бинарным векторам для паттернов строит низкоразмерные плотные вектора). Например, семантически близкие токены близки и в признаковом пространстве (табл. 5).

Во-вторых, было показано, что применение предложенной техники построения признаков для логов приводит к повышению точности решения задачи распознавания вредоносных файлов по сравнению со стандартным подходом выделения индикаторных признаков из логов (рис. 5). Более того, использование предложенных признаков совместно со стандартными приводит к еще более высокому качеству решения задачи.

Монотонные модели для онлайн классификации. Описанная выше модель с немонотонными паттернами и применением всех трех операций в пулинг слое (min, max и mean) показывает высокие результаты для классификации полных логов, однако не справляется с классификацией логов в онлайн режи-

Таблица 6: AUC-ROC для монотонных и немонотонных моделей в режиме классификации полных логов и в режиме онлайн классификации.

Режим	Немонот.	Мон. лин.	Мон. глуб.	Мон. min-max
Полные логи (AUC-ROC)	**0.999998**	0.987430	0.992089	0.993811
Онлайн (AUC-ROC)	0.511195	0.987430	0.992089	**0.993811**



(a) Чистый файл.

(b) Вредоносный файл.

Рис. 6: Предсказание монотонной и немонотонной моделей в онлайн режиме.

ме (табл. 6). Здесь под качеством работы модели в онлайн режиме понимается правильность итогового предсказания, получаемого путем выбора максимума из предсказываемых вероятностей вредоносности для всех возможных префиксов лога. Итоговое предсказание подсчитывается таким образом исходя из того, что при применении модели на практике выполнение программы будет автоматически прерываться в тот момент, когда предсказанная вероятность ее вредоносности превысит порог.

Такое низкое качество работы немонотонной модели связано с тем, что несмотря на то, что предсказание по полному логу модель чаще всего выдает верное, ее предсказания для префиксов могут сильно варьироваться внутри одного лога (рис. 6). Такое поведение объясняется двумя факторами: использованием «обеляющих» признаков и отсутствием каких-либо ограничений на промежуточные ответы для префиксов.

Для улучшения работы модели в онлайн режиме предлагается использовать монотонные по логу модели. Данная модификация может быть применена

к различным моделям [51–53], однако в данной главе мы рассмотрим модификацию нейросетевой модели, описанной выше. Для получения монотонной процедуры извлечения признаков достаточно извлекать монотонные паттерны, использовать только положительные веса в слое извлечения признаков из паттернов и только операцию max в пулинг слое (так как другие две операции приводят к немонотонным по логу признакам). Для получения монотонной нейронной сети в качестве классификатора можно использовать существующие min-max [51, 54] или lattice сети [52]. Также можно сделать обычную нейронную сеть монотонной путем использования только положительных весов и монотонных функций активации.

Эксперименты. Эксперименты проводились на большом наборе реальных логов, состоящем из примерно 4 млн. файлов, разбитых в соотношении 7:3 на обучение и контроль.

В экспериментах было показано, что качество работы монотонных моделей на задаче классификации полных логов сопоставимо с качеством немонотонных моделей. При этом в онлайн режиме монотонные модели решают поставленную задачу так же качественно, в то время как немонотонные не справляются с ней вовсе (табл. 6). Предсказания монотонной модели консистентны по времени работы программы (рис. 6) и зачастую хорошо интерпретируемы.

4. Многоклассовая модель формы со скрытыми переменными

Многоклассовая модель формы Больцмана MSBM [37] представляет собой обобщение бинарной модели формы SBM [55] на многоклассовый случай, в котором объект состоит из нескольких частей, каждой из которых ставится в соответствие свой класс. MSBM обладает большей выразительной способностью по сравнению с бинарной, так как настроить модель на вариации формы

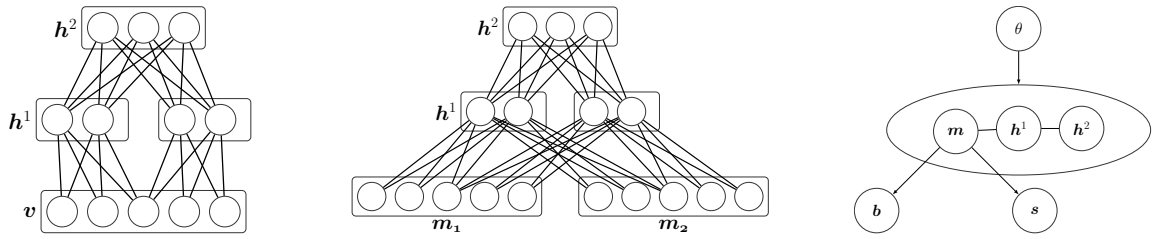
(a) (b) (c) (d)

Рис. 7: (a) – изображение, (b) – бинарная разметка b, (c) – многоклассовая разметка m, (d) семена s для 4 частей: головы, передних ног, задних ног, крупа.

отдельных частей объекта обычно проще, однако для ее обучения требуются данные с полной многоклассовой разметкой.

Нотация. Пусть нам дано D центрированных и масштабированных изображений с объектами одного типа (рис. 7a, тип: лошади в профиль). За P обозначим число частей объекта, а N — число пикселей на изображении. Пусть $B = \{b^d\}_{d=1,\dots,D}$ — множество бинарных разметок изображений (рис. 7b), где $b^d \in \{0,1\}^N$, а $M = \{m^d\}_{d=1,\dots,D}$ — множество многоклассовых разметок (рис. 7c), где $m^d \in \{0,\dots,P\}^N$, 0 соответствует фону. Также будем использовать бинарные переменные $m^d_{ip} \in \{0,1\}$ для обозначения принадлежности пикселя i на изображении d классу p и векторные переменные $m^d_p = \{m^d_{ip}\}_{i=1,\dots,N}$. Введем функцию $f_{coord}(i)$, ставящую в соответствие номеру пикселя его координаты на изображении и множество семян $S = \{s^d\}_{d=1,\dots,D}$, где s^d — это упорядоченный набор семян для изображения d, каждое семя задается своими координатами на изображении.

Модели формы. Модели формы Больцмана SBM и MSBM представляют собой трехслойные глубинные модели Больцмана (DBM, [56]) с дополнительными ограничениями. Наблюдаемый слой, соответствующий изображению, делится на 4 равные части с перекрытием и каждая часть соединяется только со своим подмножеством переменных на первом скрытом слое. Более того, веса для каждой из этих частей одинаковы. Такие ограничения позволяют существенно



(a) (b) (c)

Рис. 8: (a) – архитектура SBM, (b) – архитектура MSBM, (c) – графическая модель $p(b, s, m, h^1, h^2|\theta)$.

уменьшить число параметров глубинной модели, что позволяет избежать переобучения, ускорить процесс настройки параметров, а также обучать модель на меньшем объеме данных. Будем считать, что эти ограничения неявно учтены в соответствующей матрице весов. Архитектуры SBM и MSBM показаны на рисунке 8.

MSBM с наблюдаемым слоем m и двумя скрытыми h^1, h^2 задается с помощью распределения Гиббса:

$$p(m, h^1, h^2 \mid \theta) = \frac{1}{Z(\theta)} \exp\left(-E(m, h^1, h^2 \mid \theta)\right), \tag{13}$$

где $Z(\theta) = \sum_{m,h^1,h^2} \exp\left(-E(m, h^1, h^2 \mid \theta)\right)$ – нормировочная константа, $\theta = \{a, c^1, c^2, W^1, W^2\}$ – набор параметров модели, а энергия записывается следующим образом:

$$E(m, h^1, h^2 \mid \theta) = \sum_{i=1}^{I}\sum_{p=1}^{P} a_{i,p} m_{i,p} + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{p=1}^{P} m_{i,p} W^1_{i,j,p} h^1_j + \\ + \sum_{j=1}^{J} c^1_j h^1_j + \sum_{j=1}^{J}\sum_{k=1}^{K} h^1_j W^2_{j,k} h^2_k + \sum_{k=1}^{K} c^2_k h^2_k. \tag{14}$$

Ограничения на равенство групп весов между частями изображения и подмножествами переменных первого скрытого слоя, описанные выше, заложены в матрицу W_1. Модель SBM задается аналогичным образом.

Модели SBM и MSBM обучаются путем максимизации логарифма правдоподобия $\log p(b \mid \theta)$ с помощью EM-алгоритма. Подробное описание алгоритма

обучения приведено в [57]. Важным ограничением стандартной процедуры обучения глубинных моделей является то, что для ее применения требуются данные с полной разметкой, получение которой, особенно в многоклассовом случае, очень трудозатратная задача.

Совместная вероятностная модель. В данной главе предлагается метод обучения MSBM по данным с неполной разметкой - бинарным маскам объектов и семенам их частей (точкам на изображении, в окрестностях которых лежат пиксели заданного класса). Для этого вводится совместная вероятностная модель на бинарную маску b, многоклассовую маску m, семена s и скрытые переменные модели h^1, h^2 предполагая, что бинарная разметка b и семена s условно независимы при известной многоклассовой разметке m:

$$p(b, s, m, h^1, h^2|\theta) \;=\; p(b|m)p(s|m)p(m, h^1, h^2|\theta). \tag{15}$$

Соответствующая графическая модель представлена на рис. 8c. Под распределением $p(m, h^1, h^2|\theta)$ здесь понимается распределение Гиббса для MSBM (13).

Для задания распределения $p(b|m)$ вводится предположение о том, что при известной многоклассовой разметке m бинарная метка пикселя b_i зависит только от его же многоклассовой метки m_i, причем если пиксель принадлежит какой-либо части объекта, то он точно принадлежит объекту, иначе он принадлежит фону. Таким образом совместное распределение $p(b|m)$ задается следующим образом:

$$p(b|m) = \prod_{i=1}^{N} p(b_i|m_i) = \prod_{i=1}^{N} \left([b_i = 0][m_i = 0] + [b_i \neq 0][m_i \neq 0]\right). \tag{16}$$

Для задания распределения $p(s|m)$ вводится предположение о том, что на расположение семян все пиксели объекта влияют независимо друг от друга, а пиксели фона не влияют вовсе. Более того, на расположение семени s_{m_i}, отвечающего за часть объекта m_i, влияют только пиксели, относящиеся к этой части. Таким

образом распределение $p(s|m)$ задается следующим образом:

$$p(s|m) \propto \prod_{i:m_i \neq 0} \mathcal{N}\left(s_{m_i} | f_{coord}(i), \sigma^2\right) \propto \prod_{i:m_i \neq 0} \exp\left\{ -\frac{\|s_{m_i} - f_{coord}(i)\|_2^2}{2\sigma^2} \right\}, \quad (17)$$

где σ – внешний параметр модели. Таким образом, мы предполагаем, что пиксель что пиксель i «тянет на себя» семя m_i.

Алгоритм обучения. Для обучения параметров модели θ с помощью EM-алгоритма решается следующая задача максимизации:

$$\log P(B, S \mid \theta) = \sum_{d=1}^{D} \log p(b^d, s^d|\theta) \to \max_{\theta}. \quad (18)$$

При этом переменные b, s считаются наблюдаемыми, а m, h^1, h^2 — скрытыми.

На E-шаге находится вариационная оценка апостериорного распределения на скрытые переменные в семействе полностью факторизованных распределений

$$q^d(m, H) = \prod_{i=1}^{I} q_i^d(m_i) \prod_{j=1}^{J} q_j^d(h_j^1) \prod_{k=1}^{K} q_k^d(h_k^2) \quad (19)$$

путем минимизации дивергенции Кульбака-Лейблера:

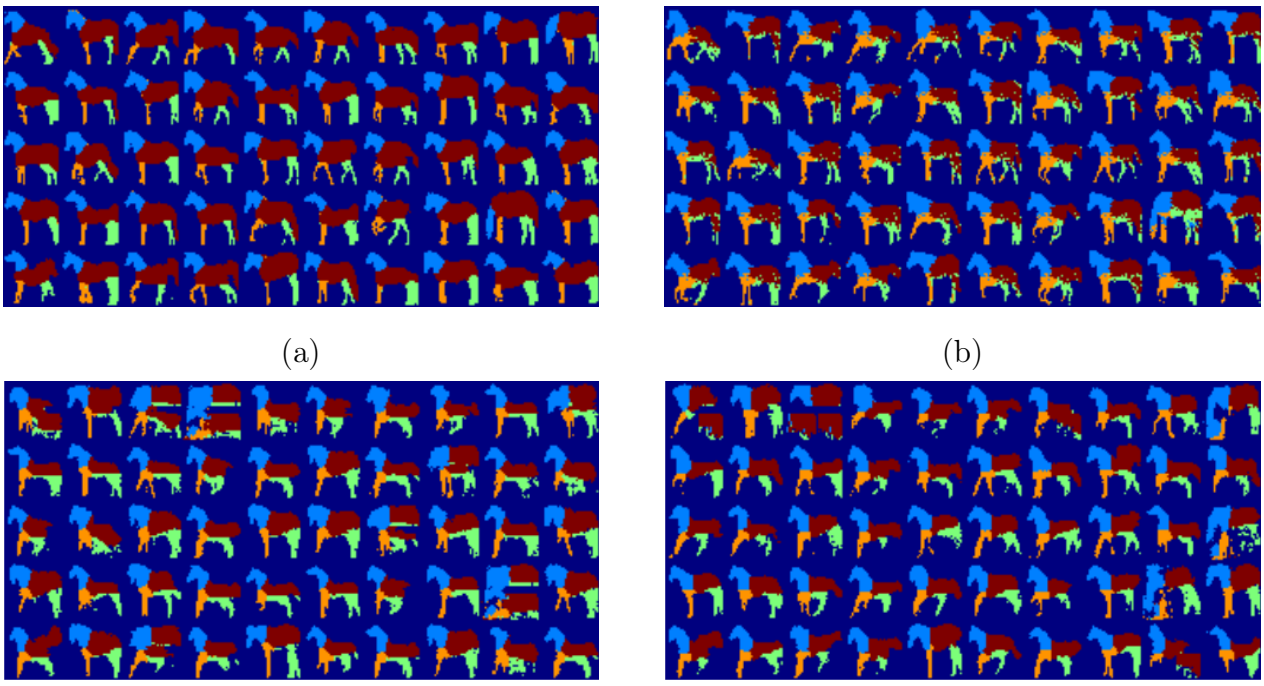
$$\min_{q^d} \mathrm{KL}\left(q^d(m, h^1, h^2) \| p(m, h^1, h^2|b^d, s^d, \theta)\right). \quad (20)$$

На M-шаге обновляются параметры модели θ путем решения следующей задачи максимизации:

$$\max_{\theta} \sum_{d=1}^{D} \left[\sum_{m,h^1,h^2} q^d(m, h^1, h^2) \log p(b^d, s^d, m, h^1, h^2|\theta) \right] \quad (21)$$

Предлагаемая процедура обучения имеет ту же вычислительную сложность, что и стандартная процедура обучения MSBM.

Использование детектора частей. Семена частей объектов на изображениях, необходимые для предлагаемой процедуры обучения MSBM, можно получать как посредством ручной разметки, так и автоматически с помощью



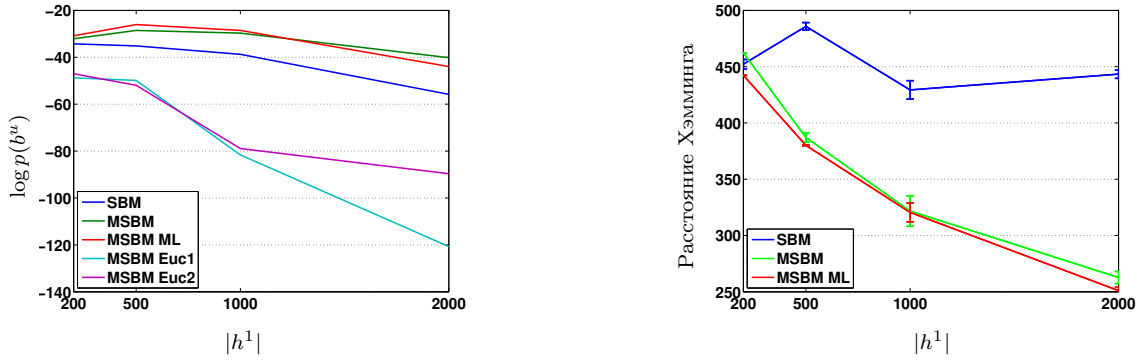
(a) (b)

(c) (d)

Рис. 9: Результаты генерации лошадей из: (a) – MSBM ML, (b) – MSBM, обученной предлагаемым методом, (c) – MSBM Euc1, (d) – MSBM Euc2.

детектора частей [36]. Процедура ручной разметки семян значительно проще получения полной многоклассовой разметки, поэтому даже в этом случае предлагаемая процедура обучения требует менее трудоемкой разметки обучающих данных. Процедура автоматического получения семян с помощью детектора частей сводится к обучению детектора частей на данных с только бинарной разметкой и ручному отбору частей необходимого типа из выделенных детектором (это достаточно сделать один раз для одного изображения).

Эксперименты. В экспериментах MSBM, обученную предлагаемым алгоритмом, сравнивается с SBM на бинарной задаче и с MSBM, обученной классическим способом, на многоклассовой задаче (MSBM ML). Также в качестве базовых моделей рассмотрены MSBM, полная многоклассовая разметка для обучения которых с помощью стандартного алгоритма получена из бинарных масок и семян с помощью простых эвристик (Euc1, Euc2). Все сравнение проводятся на двух наборах данных: лошади [14] и мотоциклы [58]. В данной работе при-



(a) (b)

Рис. 10: (a) – качество восстановления формы. Чем выше значение значение меры, тем лучше. (b) – порождение формы из семян. Расстояние Хэмминга между сгенерированными из SBM и MSBM бинарными формами и исходными, для которой брались семена (чем ниже значение значение меры, тем лучше).

водятся только результаты на первом наборе данных, результаты на втором выглядят аналогично.

В первом наборе экспериментов сравниваются результаты генерации форм из сравниваемых моделей (рис. 9) и показывается, что формы, сгенерированные из MSBM, обученной предлагаемым методом, сопоставимы по качеству с качеством сэмплов из MSBM ML (с точки зрения общей формы объектов, которая наиболее важна для дальнейшего применения моделей формы к различным задачам компьютерного зрения) и значительно превосходят по качеству сэмплы из MSBM Euc1 и MSBM Euc2.

Во втором и третьем наборах экспериментов показывается, что MSBM, обученная с помощью предлагаемого метода, показывает результаты, сравнимые с результатами для MSBM ML и превосходит модель SBM в задаче восстановления скрытой части изображения по всему остальному изображению (рис. 10a) и в задаче генерации формы объекта по известным семенам его частей (рис. 10b)

.

Положения, выносимые на защиту:

- Метод разреживания рекуррентных нейросетевых архитектур на основе

метода вариационного дропаута [21], а также его модификации для разреживания входного словаря задачи, нейронов и промежуточных гейтов в рекуррентных архитектурах. Экспериментальная валидация метода на задачах генерации последовательностей.

- Метод адаптивного времени предсказания для задач классификации последовательных данных основе метода адаптивного времени вычислений [38].

- Метод классификации последовательных данных для задач с несимметричными классами в онлайн-режиме, базирующийся на идее применения монотонной по последовательности модели. Экспериментальная валидация метода на задаче распознавания вредоносных файлов по логам их выполнения в онлайн режиме.

- Метод обучения многоклассовой больцмановской модели формы [37] по данным с только бинарной разметкой на основе вероятностной модели с латентными переменными.

Апробация результатов. Основные результаты исследования докладывались на следующих конференциях и семинарах:

1. Conference on Empirical Methods in Natural Language Processing (EMNLP), Брюссель, октябрь 2018. Тема доклада: «Bayesian Compression for Natural Language Processing». Принят постер.

2. Выездной семинар по машинному обучению НИУ ВШЭ, Вороново, апрель 2018. Тема доклада: «Байесовское разреживание рекуррентных нейронных сетей».

3. ICLR Workshop, Ванкувер, май 2018. Тема доклада: «Monotonic models for real-time dynamic malware detection». Постер.

4. Выездной семинар по машинному обучению НИУ ВШЭ, Вороново, апрель 2018. Тема доклада: «Monotonic models for real-time dynamic malware detection».

5. Workshop on Learning to Generate Natural Language, ICML, 2017, Сидней, август 2017. Тема доклада: «Bayesian Sparsification of Recurrent Neural Networks». Постер.

6. Выездной семинар по машинному обучению НИУ ВШЭ, Вороново, май 2017. Тема доклада: «Байесовская регуляризация рекуррентных нейронных сетей».

7. ICLR Workshop, Тулон, апрель 2017. Тема доклада: «Semantic embeddings for program behavior». Постер.

8. Выездной семинар по машинному обучению НИУ ВШЭ, Вороново, май 2017. Тема доклада: «Machine learning for malware detection».

9. Рождественский коллоквиум по компьютерному зрению, Сколтех, Москва, декабря 2016. Тема доклада: «Deep Part-Based Generative Shape Model with Latent Variables».

10. British Machine Vision Conference (BMVC), Йорк, сентябрь 2016. Тема доклада: «Deep Part-Based Generative Shape Model with Latent Variables». Постер.

11. Всероссийская конференция с международным участием Математические методы распознавания образов, г. Светлогорск Калининградской области, сентябрь 2015. Тема доклада: «Многоклассовая модель формы со скрытыми переменными».

12. 24-я международная конференция по компьютерной графике и зрению,

ГрафиКон'2014, Ростов-на-Дону, октябрь 2014. Тема доклада: «Многоклассовая модель формы со скрытыми переменными».

Публикации. Материалы исследования опубликованы в работах, перечисленных ниже. Из перечисленных статей две индексируются в базе Scopus (1,6) и еще одна (7) опубликована в журнале, входящем в перечень ВАК.

1. Chirkova N., Lobacheva E., Vetrov D. Bayesian Compression for Natural Language Processing // Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistic 2018. Scopus.

2. Maksim Ryabinin, Ekaterina Lobacheva. Adaptive prediction time for sequence classification. Препринт. 2018.

3. Chistyakov A., Lobacheva E., Shevelev A., Romanenko A. Monotonic models for real-time dynamic malware detection // ICLR Workshop, 2018.

4. Ekaterina Lobacheva, Nadezhda Chirkova, Dmitry Vetrov. Bayesian Sparsification of Recurrent Neural Networks // Workshop on Learning to Generate Natural Language, ICML, 2017.

5. Chistyakov A., Lobacheva E., Kuznetsov A., Romanenko A. Semantic embeddings for program behavior // ICLR Workshop, 2017.

6. Alexander Kirillov, Mikhail Gavrikov, Ekaterina Lobacheva, Anton Osokin and Dmitry Vetrov. Deep Part-Based Generative Shape Model with Latent Variables // Proceedings of the 27th British Machine Vision Conference. 2016. Scopus.

7. Кириллов А. Н., Гавриков М. И., Лобачева Е. М., Осокин А. А., Ветров Д. П. Многоклассовая модель формы со скрытыми переменными // Интеллектуальные системы. Теория и приложения. 2015. Т. 19. №2. С. 75-95. ВАК.

Личный вклад автора. Метод разреживания рекуррентных нейронных сетей разработан в соавторстве с Чирковой Н.А., его экспериментальная валидация на задачах генерации последовательностей проведена автором лично. Метод адаптивного времени предсказания разработан автором лично, его экспериментальная валидация проведена совместно с Рябининым М.К. Монотонный метод классификации последовательных данных в онлайн режиме, а также метод построения признакового описания для логов на основе их графовой структуры разработаны совместно с Чистяковым А.С., их экспериментальная валидация проведена совместно с Чистяковым А.С. и Шевелевым А.С. Метод обучения многоклассовой больцмановской модели формы разработан в соавторстве с Кирилловым А.Н. и Гавриковым М.И. Вклад остальных соавторов заключается в рецензировании программного кода экспериментов, технической помощи в постановке экспериментов, обсуждениях полученных результатов, правках текста статей, постановке решаемой задачи и общем руководстве исследованиями.

Список литературы

1. Redmon Joseph, Farhadi Ali. YOLO9000: Better, Faster, Stronger // CoRR. 2016. Vol. abs/1612.08242.
2. Chen Liang-Chieh, Papandreou George, Schroff Florian, Adam Hartwig. Rethinking Atrous Convolution for Semantic Image Segmentation // CoRR. 2017. Vol. abs/1706.05587.
3. Ren Shaoqing, He Kaiming, Girshick Ross B., Sun Jian. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks // CoRR. 2015. Vol. abs/1506.01497.
4. Badrinarayanan Vijay, Kendall Alex, Cipolla Roberto. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation // CoRR. 2015. Vol. abs/1511.00561.
5. Zhou Bolei, Lapedriza Agata, Xiao Jianxiong et al. Learning Deep Features for Scene Recognition using Places Database // Advances in Neural Information Processing Systems 27. Curran Associates, Inc., 2014. P. 487–495.
6. Ha David, Dai Andrew, Le Quoc V. HyperNetworks // Proceedings of the International Conference on Learning Representations (ICLR). 2017.
7. Wu Yonghui, Schuster Mike, Chen Zhifeng et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation // Computing Research Repository. 2016. Vol. arXiv:1609.08144.
8. Ren Mengye, Kiros Ryan, Zemel Richard S. Exploring Models and Data for Image Question Answering // Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada. 2015. P. 2953–2961.
9. Chan William, Jaitly Navdeep, Le Quoc V., Vinyals Oriol. Listen, At-

tend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition // Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2016.

10. Le Quoc V., Jaitly Navdeep, Hinton Geoffrey E. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units // Computing Research Repository. 2015. Vol. arXiv:1504.00941.
11. Tjandra Andros, Sakti Sakriani, Nakamura Satoshi. Compressing Recurrent Neural Network with Tensor Train // Computing Research Repository. 2017. Vol. arXiv:1705.08052.
12. Narang Sharan, Diamos Gregory F., Sengupta Shubho, Elsen Erich. Exploring Sparsity in Recurrent Neural Networks // Proceedings of the International Conference for Learning Representations (ICLR). 2017.
13. Wen Wei, He Yuxiong, Rajbhandari Samyam et al. Learning Intrinsic Sparse Structures within Long Short-Term Memory // International Conference on Learning Representations. 2018.
14. Borenstein Eran. Combining top-down and bottom-up segmentation // In Proceedings IEEE workshop on Perceptual Organization in Computer Vision, CVPR. 2004. P. 46.
15. Van Ginneken B., Frangi A. F., Staal J. J. et al. Active shape model segmentation with optimal features // IEEE Transactions on Medical Imaging. 2002. Vol. 21, no. 8. P. 924–933.
16. Chan Tony F., Shen Jianhong. Non-Texture Inpainting by Curvature-Driven Diffusions (CDD) // J. Visual Comm. Image Rep. 2001. Vol. 12. P. 436–449.
17. Ferrari V., Jurie F., , Schmid C. From Images to Shape Models for Object Detection // International Journal of Computer Vision. 2010. Vol. 87, no. 3. P. 284–303.
18. Lecun Yann, Bottou Léon, Bengio Yoshua, Haffner Patrick. Gradient-based learning applied to document recognition // Proceedings of the

IEEE. 1998. P. 2278–2324.

19. Hochreiter Sepp, Schmidhuber Jürgen. Long Short-Term Memory // Neural Comput. 1997. — . Vol. 9, no. 8. P. 1735–1780.
20. Gilmer Justin, Schoenholz Samuel S., Riley Patrick F. et al. Neural Message Passing for Quantum Chemistry // Proceedings of the 34th International Conference on Machine Learning. Vol. 70 of Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 2017. — 06–11 Aug. P. 1263–1272.
21. Molchanov Dmitry, Ashukha Arsenii, Vetrov Dmitry. Variational Dropout Sparsifies Deep Neural Networks // Proceedings of the 34th International Conference on Machine Learning, ICML 2017. 2017.
22. Neklyudov Kirill, Molchanov Dmitry, Ashukha Arsenii, Vetrov Dmitry P. Structured Bayesian Pruning via Log-Normal Multiplicative Noise // Advances in Neural Information Processing Systems 30. 2017. P. 6778–6787.
23. Louizos Christos, Ullrich Karen, Welling Max. Bayesian Compression for Deep Learning // Advances in Neural Information Processing Systems 30. 2017. P. 3288–3298.
24. Yu Adams Wei, Lee Hongrae, Le Quoc V. Learning to Skim Text // Annual Meeting of the Association for Computational Linguistics. 2017.
25. Yeung Serena, Russakovsky Olga, Mori Greg, Fei-Fei Li. End-to-end Learning of Action Detection from Frame Glimpses in Videos // IEEE Conference on Computer Vision and Pattern Recognition. 2016.
26. Campos Víctor, Jou Brendan, i Nieto Xavier Giró et al. Skip RNN: Learning to Skip State Updates in Recurrent Neural Networks // International Conference on Learning Representations. 2018.
27. Aliakbarian Sadegh, Saleh Fatemeh, Salzmann Mathieu et al. Encouraging LSTMs to Anticipate Actions Very Early // IEEE International Conference on Computer Vision (ICCV). 2017. — 10. P. 280–289.
28. Wang Wenlin, Chen Changyou, Wang Wenqi et al. Earliness-Aware Deep

Convolutional Networks for Early Time Series Classification. 2016.

29. Bayer Ulrich, Comparetti Paolo Milani, Hlauschek Clemens et al. Scalable, Behavior-Based Malware Clustering. // NDSS / Citeseer. Vol. 9. 2009. P. 8–11.
30. Berlin Konstantin, Slater David, Saxe Joshua. Malicious behavior detection using windows audit logs // Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security / ACM. 2015. P. 35–44.
31. Huang Wenyi, Stokes Jack W. MtNet: a multi-task neural network for dynamic malware classification // Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2016. P. 399–418.
32. Salehi Zahra, Sami Ashkan, Ghiasi Mahboobe. MAAR: Robust features to detect malicious activity based on API calls, their arguments and return values // Engineering Applications of Artificial Intelligence. 2017. Vol. 59. P. 93–102.
33. Pascanu Razvan, Stokes Jack W., Sanossian Hermineh et al. Malware classification with recurrent networks // IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) / IEEE. 2015. P. 1916–1920.
34. Kolosnjaji Bojan, Zarras Apostolis, Webster George, Eckert Claudia. Deep Learning for Classification of Malware System Call Sequences // Australasian Joint Conference on Artificial Intelligence / Springer. 2016. P. 137–149.
35. Yangel B., Vetrov D. Image Segmentation with a Shape Prior Based on Simplified Skeleton // Energy Minimization Methods in Computer Vision and Pattern Recognition. Vol. 6819 of Lecture Notes in Computer Science. 2011. P. 247–260.
36. Felzenszwalb P. F., Girshick R. B., McAllester D., Ramanan D. Object Detection with Discriminatively Trained Part Based Models // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010. Vol. 32, no. 9.

P. 1627–1645.

37. Eslami S. M. Ali, Williams Chris. A Generative Model for Parts-based Object Segmentation // Advances in Neural Information Processing Systems 25. 2012. P. 100–107.
38. Graves Alex. Adaptive Computation Time for Recurrent Neural Networks // CoRR. 2016. Vol. abs/1603.08983.
39. Kingma Diederik P, Salimans Tim, Welling Max. Variational Dropout and the Local Reparameterization Trick // Advances in Neural Information Processing Systems 28. 2015. P. 2575–2583.
40. Kingma Diederik P, Welling Max. Auto-Encoding Variational Bayes // Proceedings of the International Conference for Learning Representations (ICLR). 2014.
41. Gal Yarin, Ghahramani Zoubin. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks // Advances in Neural Information Processing Systems 29 (NIPS). 2016.
42. Fortunato Meire, Blundell Charles, Vinyals Oriol. Bayesian Recurrent Neural Networks // Computing Research Repository. 2017. Vol. arXiv:1704.02798.
43. Hochreiter Sepp, Schmidhuber Jürgen. Long Short-Term Memory // Neural Comput. 1997. — . Vol. 9, no. 8. P. 1735–1780.
44. Marcus Mitchell P., Marcinkiewicz Mary Ann, Santorini Beatrice. Building a Large Annotated Corpus of English: The Penn Treebank // Comput. Linguist. 1993. — . Vol. 19, no. 2. P. 313–330.
45. Mikolov T., Kombrink S., Burget L. et al. Extensions of recurrent neural network language model // 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2011. — May. P. 5528–5531.
46. Bahdanau Dzmitry, Cho Kyunghyun, Bengio Yoshua. Neural machine translation by jointly learning to align and translate // International Conference on Learning Representations. 2015.

47. LeCun Yann, Bottou Léon, Bengio Yoshua, Haffner Patrick. Gradient-based learning applied to document recognition // Proceedings of the IEEE. 1998. Vol. 86, no. 11. P. 2278–2324.
48. Soomro Khurram, Zamir Amir Roshan, Shah Mubarak. UCF101: A dataset of 101 human actions classes from videos in the wild // arXiv preprint arXiv:1212.0402. 2012.
49. Keyi Yu Alexander G. Schwing Jian Peng, Yang Liu. Fast and Accurate Text Classification: Skimming, Rereading and Early Stopping. 2018.
50. Kolosnjaji Bojan, Zarras Apostolis, Webster George, Eckert Claudia. Deep Learning for Classification of Malware System Call Sequences // Australasian Joint Conference on Artificial Intelligence / Springer. 2016. P. 137–149.
51. Sill Joseph. Monotonic Networks // NIPS. 1997.
52. You Seungil, Ding David, Canini Kevin et al. Deep Lattice Networks and Partial Monotonic Functions // NIPS. 2017.
53. Potharst R., Feelders A. J. Classification Trees for Problems with Monotonicity Constraints // ACM SIGKDD Explorations Newsletter. 2002. — . Vol. 4, no. 1. P. 1–10.
54. Daniels Hennie, Velikova Marina. Monotone and Partially Monotone Neural Networks // IEEE Transactions on Neural Networks. 2010. — . Vol. 21, no. 6. P. 906–917.
55. Eslami S. M. Ali, Heess Nicolas, Williams Christopher K. I., Winn John. The Shape Boltzmann Machine: a Strong Model of Object Shape // International Journal of Computer Vision. 2013.
56. Hinton G. E., Osindero S., Teh Y. W. A fast learning algorithm for deep belief nets // Neural Computation. 2006. Vol. 18, no. 7. P. 1527–1554.
57. Salakhutdinov R., Hinton G. An Efficient Learning Procedure for Deep Boltzmann Machines // Neural Computation. 2012. Vol. 24, no. 8. P. 1967–2006.

58. Fei-Fei L., Fergus R., Perona P. One-shot learning of object categories // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2006. Vol. 28(4). P. 594–611.