DOI 10.37882/2223–2966.2021.05–2.16

ОПТИМИЗАЦИОННЫЕ МЕТОДЫ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РЕШЕНИЯ ЗАДАЧИ БИНАРНОЙ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

Крутов Тимофей Юрьевич
МГТУ им. Н.Э. Баумана
timofeykrutov@gmail.com
Афанасьев Геннадий Иванович
К.т.н., доцент, МГТУ им. Н.Э. Баумана
gaipcs@bmstu.ru
Нестеров Юрий Григорьевич
К.т.н., доцент, МГТУ им. Н.Э. Баумана
ugn@bmstu.ru

Аннотация. Проведен сравнительный анализ существующих методов оптимизации нейронных сетей и определен ряд оптимизаторов, показывающих наилучшее качество обучения для решения задачи бинарной классификации в распознавании изображений используемого набора данных. Рассмотрен метод градиентного спуска, SGD, метод Нестерова и Импульсный метод. Описаны адаптивные методы оптимизации такие как Adagrad, RMSprop и Adam. Рассмотрены две архитектуры нейронных: первая архитектура представляет собой сверточную нейронную сеть с четырьмя слоями свёртки, вторая сеть состоит из предобученной на наборе ImageNet нейронной сети VGG19 с добавленным классификатором. Состав слоев сети описан в тексте и на рисунках. В качестве обучающего набора использовался набор данных «Dogs vs. Cats» со сбалансированными классами изображений. Результаты обучения и тестов моделей приведены на графиках точности и потерь. Описаны рекомендации по выбору архитектуры нейронных сетей.

Ключевые слова: оптимизаторы нейронных сетей, перенос обучения, модели нейронных сетей, свёрточные нейронные сети, VGG19.

OPTIMIZATION METHODS OF NEURAL NETWORKS FOR SOLVING THE PROBLEM OF BINARY CLASSIFICATION

T. Krutov
G. Afanasyev
Yu. Nesterov

Summary. The aim of this work is to carry out a comparative analysis of existing methods for optimizing neural networks and to determine the best optimizers for solving the problem of binary classification in pattern recognition. A comparative analysis of existing methods for optimizing neural networks is carried out and a number of optimizers are identified that show the best quality of training for solving the problem of binary classification in image recognition of the used data set. The article briefly describes mathematical expressions for calculating the updated parameters of a neural network. The gradient descent method, SGD, the Nesterov method, and the Momentum method are considered. Adaptive optimization methods such as Adagrad, RMSProp, and Adam are also described. Two neural architectures are considered: the first architecture is a convolutional neural network with four convolution layers, the second network consists of a VGG19 neural network pre-trained on an ImageNet set with an added classifier. Additional training of the model is performed by freezing all layers of the VGG19 neural network except for the layers starting with the block5_conv1 layer. The composition of the network layers is described in the text and in the figures. The "Dogs vs. Cats" dataset with balanced image classes was used as a training set. The models were trained on the CPU without using graphics accelerators. The results of training and testing models are shown in the graphs of accuracy and loss. For ease of perception, each graph contains the learning curves of all models. Additionally, a boxplot diagram is constructed showing the probability distribution and median estimates on the test data set. Recommendations for choosing the architecture of neural networks are described.

Keywords: neural network optimizers, transfer learning, neural network models, convolutional neural networks, VGG19.

Введение

Важнейшим звеном нейронной сети является оптимизатор. Именно он выполняет задачу нахождения минимума функции ошибки и обучения нейросети. Оптимизатор и его параметры определяют качество обучения модели и влияет на результаты, получающиеся на тестовом наборе данных. Для разных наборов данных и задач показатели обучения оптимизаторами будут отличаться, поэтому выбор метода оптимизации является существенным этапом построения модели нейронной сети.

***

Одним из основных и наиболее простых алгоритмов оптимизации является алгоритм градиентного спуска. Реализация метода предполагает оптимизацию путем вычисления градиента функции ошибок с последующим пересчетом весовых коэффициентов нейронной сети. Движение на каждой последующей итерации в этом случае осуществляется в направлении антиградиента, то есть в направлении спуска к минимуму функции. Математически алгоритм градиентного спуска можно описать следующим образом:

$$\theta_k = \theta_{k-1} - \delta\nabla\mathcal{F}(\theta_{k-1}),$$

где θ_k — весовые коэффициенты на шаге k, δ — скорость обучения модели, $\nabla\mathcal{F}$ — градиент функции ошибок.

Функция ошибок градиентного спуска рассчитывается в виде:

$$\mathcal{F}(\omega) = \sum\mathcal{F}\,(f(x,\omega),y),$$

где $\mathcal{F}(f(x,\omega),y)$ — функция ошибки предсказания нейронной сети с весами ω и набором признаков x, вычисленная в виде разницы значений спрогнозированного моделью результата $f(x,\ \omega)$ и фактического y. С точки зрения вычислений градиентный спуск очень затратный алгоритм, так как вычисление градиента производится на всем обучающем наборе. В связи с этим в явном виде для оптимизации нейронных сетей он используется достаточно редко [1].

На практике более подходящим является алгоритм стохастического градиентного спуска SGD (stochastic gradient descent). Этот алгоритм является модернизацией простого градиентного спуска и вместо вычисления градиента для всего обучающего набора использует перерасчёт весов по конкретному примеру обучающего набора [2]. Однако возможно использование пакетного стохастического градиентного спуска, в котором применяется обучение по мини-батчам. Разбиение на пакеты позволяет существенно повысить скорость обучения модели за счёт варьируемости размера мини-батчей.

$$\theta_k = \theta_{k-1} - \delta\nabla\mathcal{F}((f(x_k,\theta_{k-1}),y_k))$$

Требуется отметить, что в алгоритмах градиентного и стохастического градиентного спуска скорость обучения задается пользователями перед запуском алгоритма и определяется гиперпараметром δ. Этот гиперпараметр задает шаг алгоритма обучения, что отражает скорость сходимости к минимуму функции ошибок. Шаг не изменяется во время обучения модели. Слишком большие значения δ могут привести к тому, что глобальный минимум функции так и не будет найден. Слишком маленький шаг приводит к увеличению времени обучения нейронной сети.

В импульсном методе [3] вводится коэффициент γ, определяющий степень влияния предыдущего значения градиента функции ошибок. Значение параметра γ не превышает единицы [4].

$$v_{k+1} = \gamma v_k + \delta\nabla\mathcal{F}(\theta)$$

$$\theta_{k+1} = \theta_k - v_{k+1}$$

Метод Нестерова [5] является вариантом использования импульсного метода и рассчитывает значение градиента функции ошибок в точке $\theta - \gamma v_k$.

$$v_{k+1} = \gamma v_k + \delta\nabla\mathcal{F}(\theta - \gamma v_k)$$

$$\theta_{k+1} = \theta_k - v_{k+1}$$

Существуют более сложные методы адаптивной оптимизации. Один из таких методов — Adagrad (adaptive gradient) [6]. Метод заключается в применении идеи, что шаг изменения должен быть меньше у тех параметров, которые в большей степени варьируются в данных [1]. Вычисляются частные производные по каждому параметру, и при малых значениях частной производной скорость обучения изменяется медленно. Для реализации этой идеи применяется матрица сумм квадратов градиентов каждого параметра:

$$G_k = G_k + g_k^2\,,\ g_k = \nabla\mathcal{F}(\theta)$$

Таким образом для наиболее часто используемых параметров нейронной сети коэффициент G_k будет постоянно накапливать значение.

Функция обновления параметров метода Adagrad выглядит следующим образом:

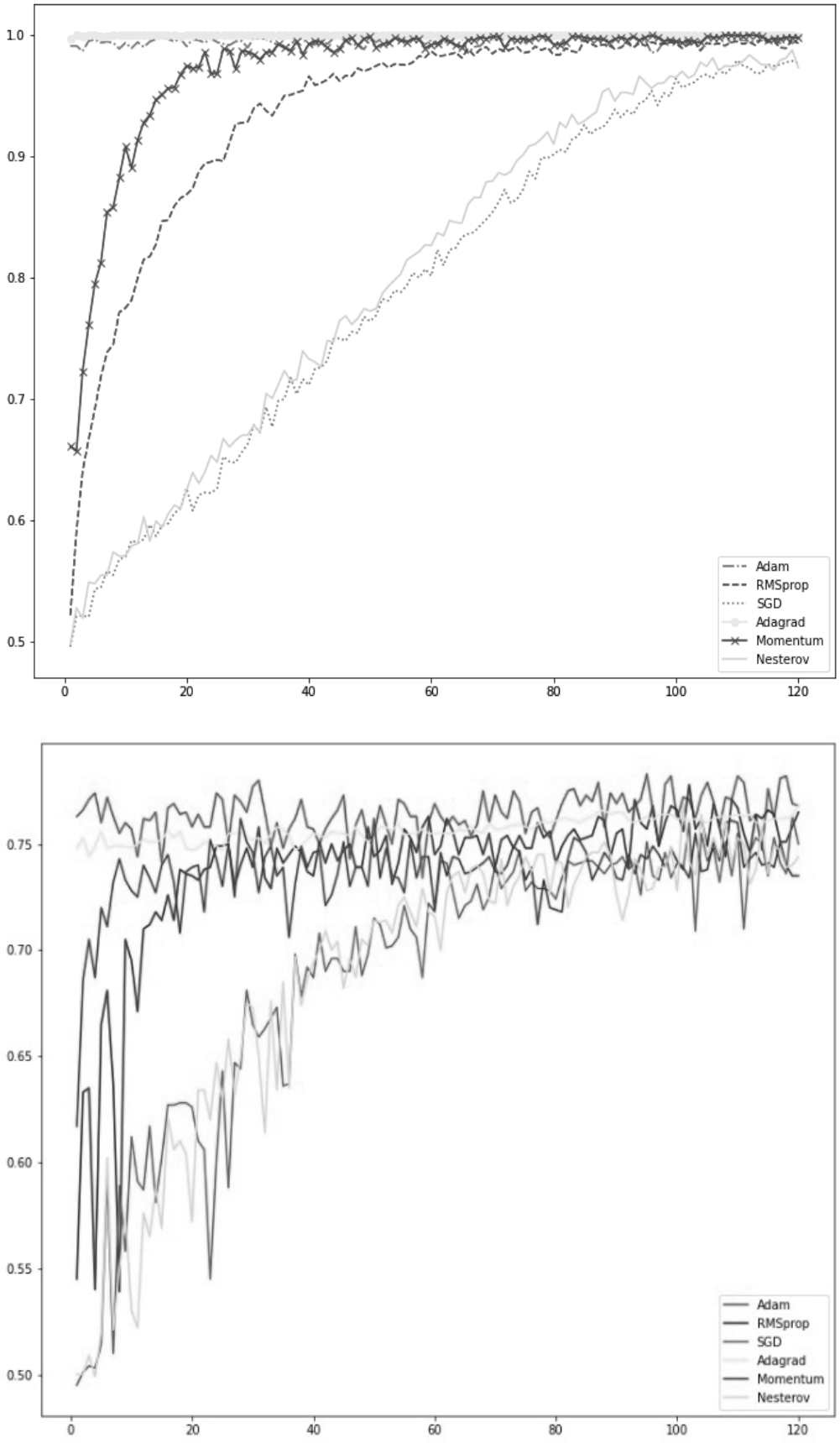

Рис. 1. Точность обучения (сверху), валидационная точность (снизу)

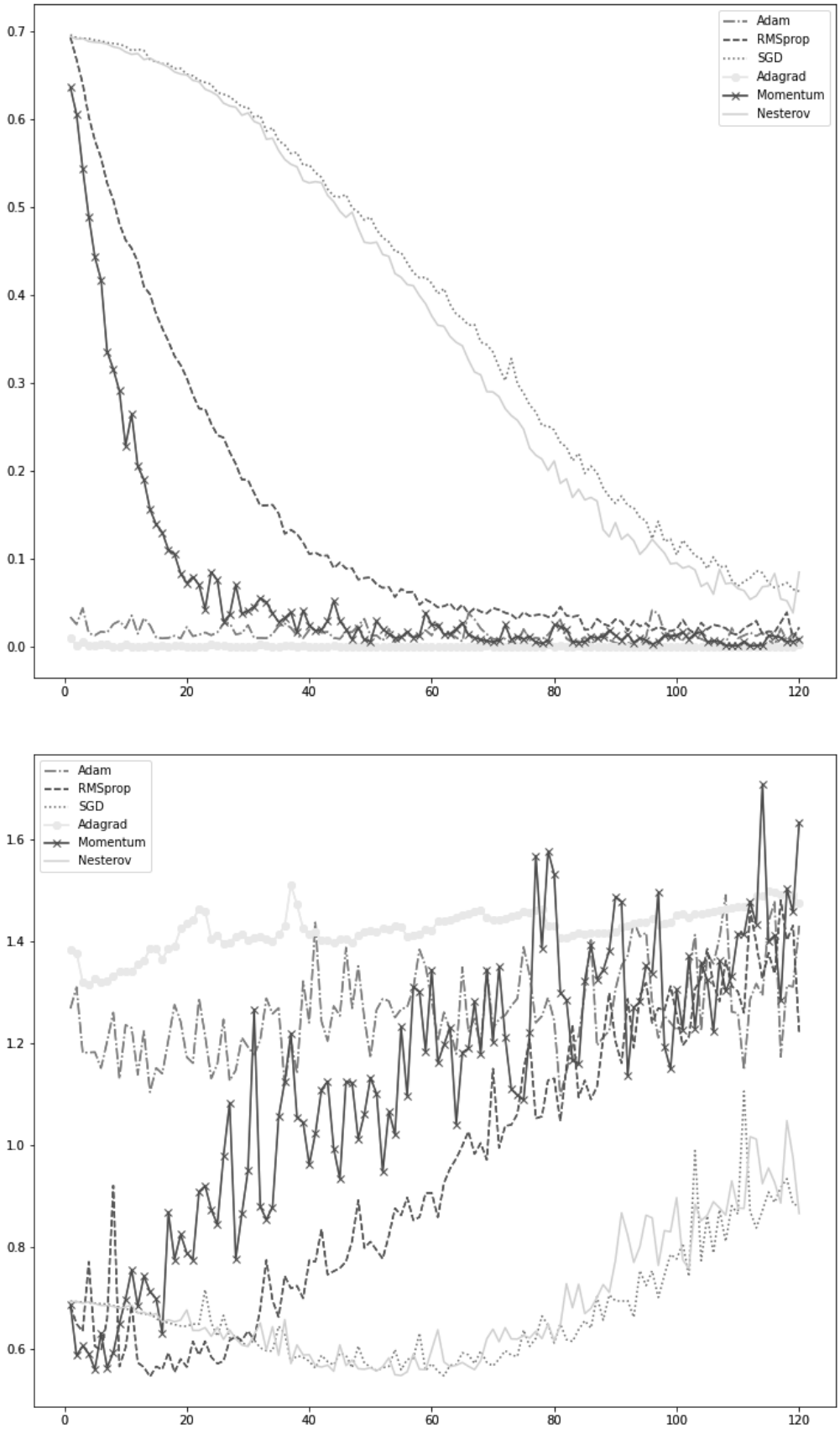

Рис. 2. Потери на этапе обучения (сверху), потери на валидационном наборе (снизу)

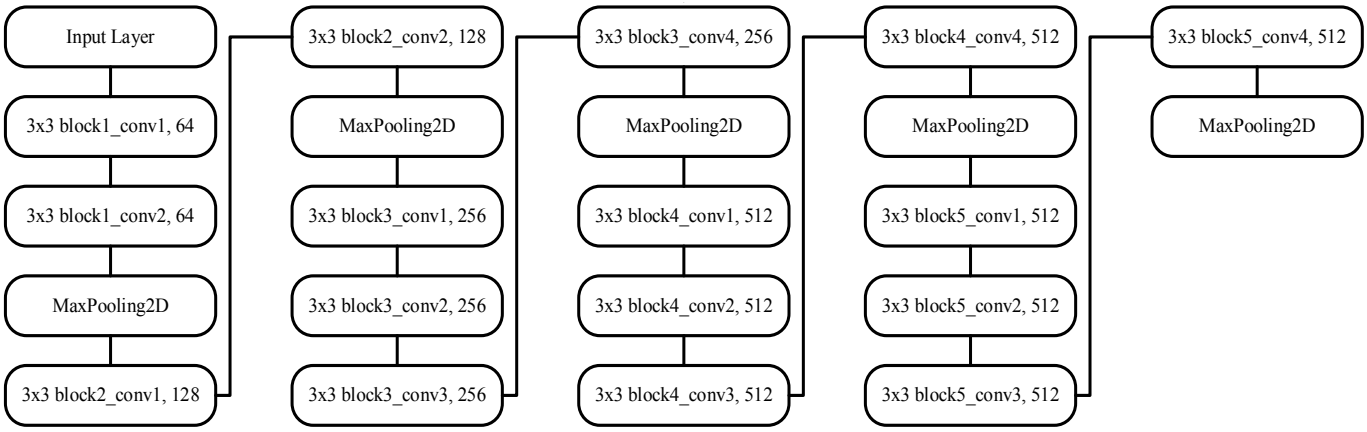

Рис. 3. Архитектура нейронной сети VGG19

$$\theta_{k+1} = \theta_k - \frac{\delta}{\sqrt{G_k + \epsilon}} \cdot g_k,$$

где ϵ — сглаживающий параметр, обычно, лежащий в диапазоне от 10^{-6} до 10^{-8}.

Другим популярным методом является RMSprop [7] (root mean squares propagation). Оба оптимизатора решают проблему постоянного увеличения матрицы квадратов градиентов G_k, так как это приводит к минимальным обновлениям параметров и возможной остановке обучения.

$$E[g^2]_k = \rho E[g^2]_{k-1} + (1-\rho) g_k^2$$

$$\theta_{k+1} = \theta_k - \frac{\delta}{\sqrt{E[g^2]_k + \epsilon}} g_k,$$

где $E[g^2]k$ — скользящее среднее в момент времени k, ρ — метапараметр истории скользящего среднего.

Оптимизационный метод Adam (adaptive moment estimation), предложенный в 2015 году в статье [8], использует сглаженные версии среднего и среднеквадратичного градиентов:

$$m_k = \beta_1 m_{k-1} + (1-\beta_1) g_k$$

$$v_k = \beta_2 m + (1-\beta_2) g_k^2$$

$$\theta_{k+1} = \theta_k - \frac{\delta}{\sqrt{v_t + \epsilon}} m_t$$

Экспериментальная часть и результаты

В рамках работы выполняется поиск наиболее точного алгоритма оптимизации модели машинного обучения для решения задачи классификации графических изображений. При решении этой задачи использовался свободно распространяемый набор данных «Dogs vs. Cats» [9].

Все вычисления производились на персональном компьютере с нижеперечисленными техническими характеристиками:

- Центральный процессор Intel Core i5–3550, 3500 МГц
- Оперативная память DDR3 8 Гб

В качестве модели нейронной сети использовалась свёрточная нейронная сеть со следующей архитектурой слоёв:

- Conv2D(32, (3,3), activation = 'relu')
- MaxPooling2D(2,2)
- Conv2D(64, (3,3), activation = 'relu')
- MaxPooling2D(2,2)
- Conv2D(128,(3,3), activation = 'relu')
- MaxPooling2D(2,2)
- Conv2D(128,(3,3), activation = 'relu')
- MaxPooling2D(2,2)
- Flatten
- Dropout(0.5)
- Dense(512, activation = 'relu')
- Dense(1, activation = 'sigmoid')

Обучение модели производилось на CPU, без привлечения графических ускорителей.

Далее в виде графиков (рисунок 1, рисунок 2) приведены результаты, полученные после построения и обучения модели свёрточной нейронной сети.

Можно выделить методы Adam и Adagrad как обладающие наискорейшей сходимостью. Наименьшая скорость сходимости наблюдается у стохастического градиентного спуска и метода Нестерова.

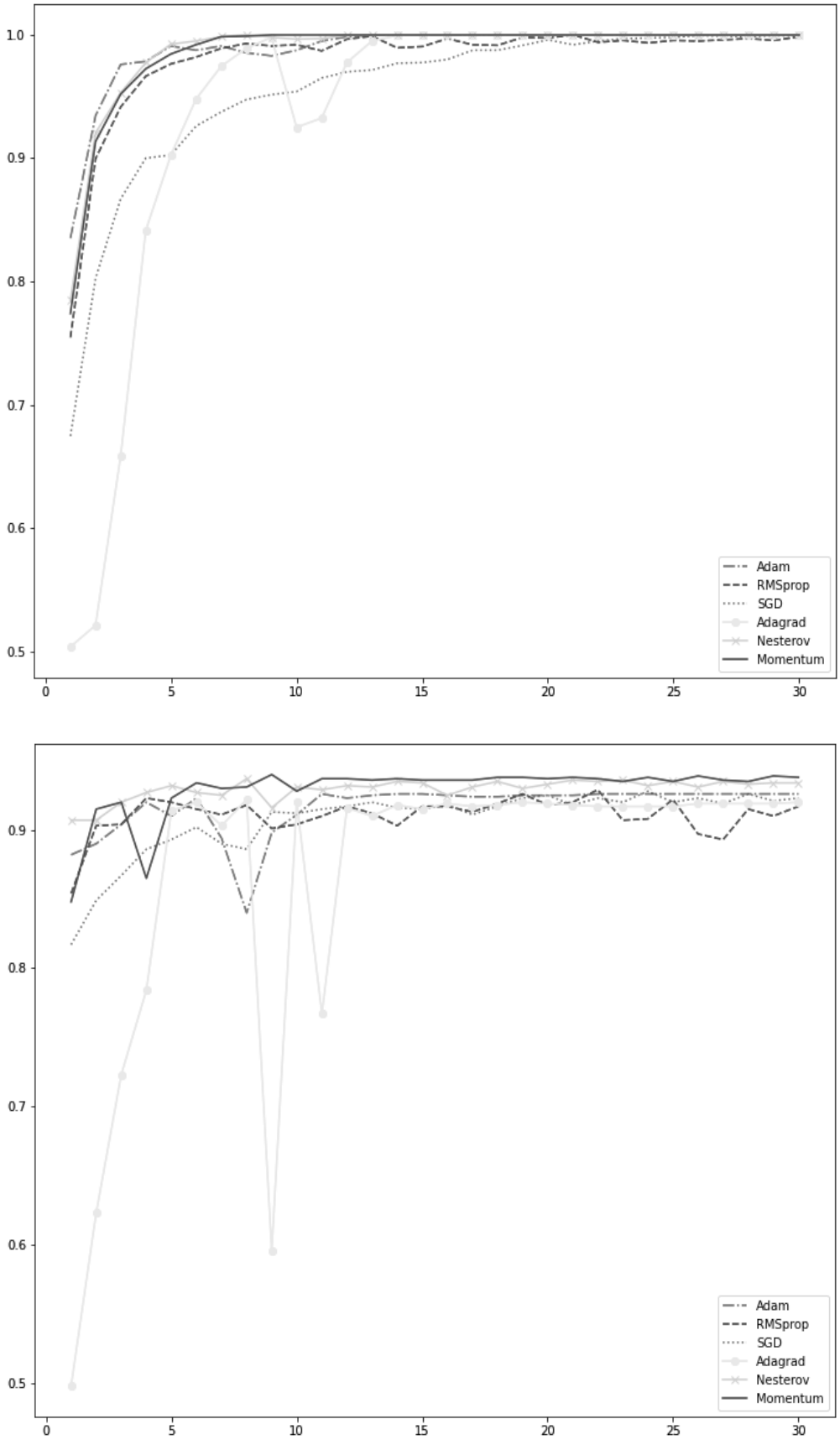

Рис. 4. Точность обучения (сверху), валидационная точность (снизу)

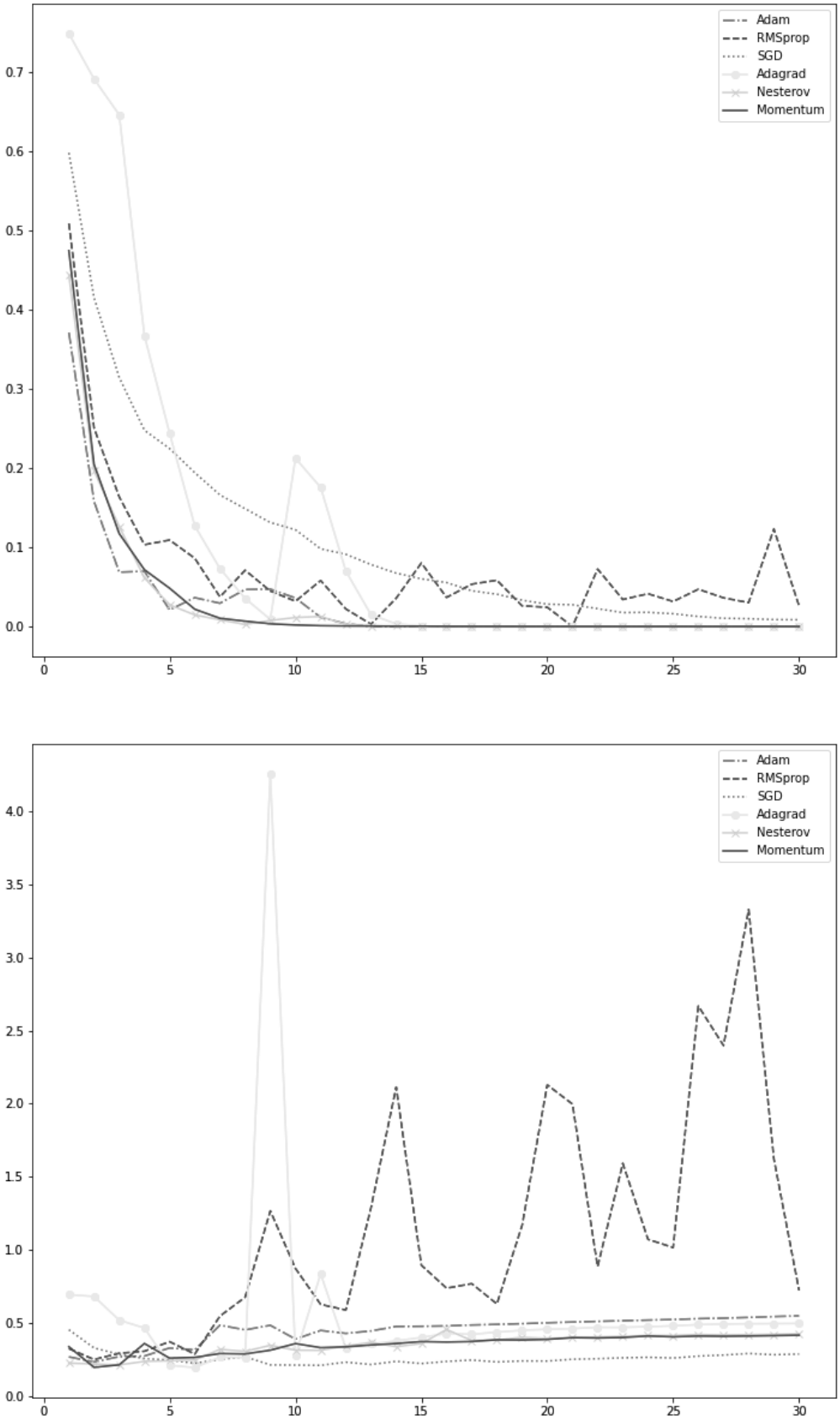

Рис. 5. Потери на этапе обучения (сверху), потери на валидационном наборе (снизу)

Таблица 1. Результаты точности моделей

Accuracy	Adam	RMSprop	SGD	Adagrad	Nesterov	Momentum
CNN	0,7829	0,7779	0,763	0,768	0,762	0,7689
VGG19	0,9259	0,929	0,9279	0,9219	0,9369	0,9399

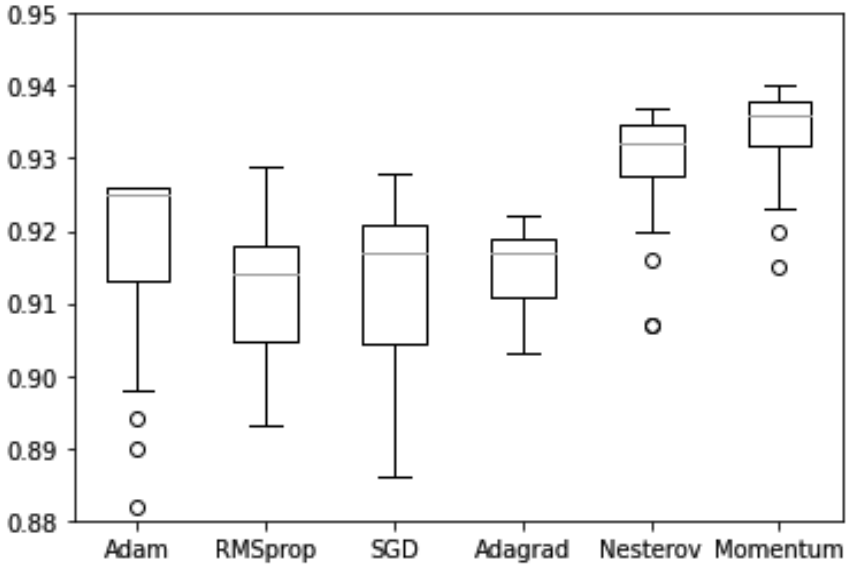

Рис. 6. Диаграмма размаха точности тестового набора данных в зависимости от используемого оптимизатора

Как видно из приведенных результатов, такая архитектура модели вне зависимости от используемого оптимизатора не дает хорошего качества обучения. На обучающем наборе наблюдается точность 0,97–1, в то время как на тестовом наборе точность распознавания не поднимается выше 0,77.

Для улучшения точности распознавания используется предобученная сеть VGG19 [9], архитектура которой приведена на рисунке 3.

К предобученной сети VGG19 были добавлены следующие слои:

- Flatten
- Dropout(0,5)
- Dense(512, activation='relu')
- Dense(1, activation='sigmoid')

Для дообучения нейронной сети на рассматриваемом наборе данных были заблокированы для обучения все слои нейронной сети до block5_conv1. Начиная со слоя block5_conv1, производилось обучение параметров сети.

Применение технологии переноса обучения (transfer learning) позволило существенно повысить качество модели при сохранении весов предобученной модели [10, 11]. Для случая бинарной классификации при дообучении верхних слоёв сети VGG19 и полносвязного слоя были получены следующие результаты:

На рисунке 4 видно, что наибольшая точность модели достигается при использовании импульсного метода. Чуть меньшую точность демонстрирует оптимизационный метод Нестерова. График функции потерь приведен на рисунке 5.

Для наглядности дополнительно построена диаграмма размаха (рис. 6), чтобы продемонстрировать распределение вероятностей точности и оценки медианы на тестовом наборе данных. Некоторые выбросные значения распределений были удалены для обеспечения верного восприятия масштаба диаграмм.

Достигнутые результаты точности двух используемых моделей на тестовых данных приведены в Таблице 1.

Выводы

По полученным результатам можно судить о том, что используемые методы нейросетевой оптимизации в задаче бинарной классификации изображений для рассматриваемого набора данных достаточно близки друг к другу по качеству распознавания. Различия между ними составляет 1–2%, что не всегда является существенным. Куда более важную роль играет архитектура нейронной сети. Так при использовании простой сети точность на тестовых данных не превышала 79%, в то время как в модели с использованием предобученной сети точность составила почти 94%. Именно поэтому для данного типа задач рекомендуется использовать архитектуру нейронной сети, основанную на переносе обучения.

ЛИТЕРАТУРА

1. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей — СПб.: Питер, 2020. — 480 с.
2. Чио К., Фримэн Д. Машинное обучение и безопасность — М.: ДМК Пресс, 2020. — 388 с.
3. Поляк Б.Т. О некоторых способах ускорения сходимости итерационных методов. / Б.Т. Поляк // Журнал вычислительной математики и математической физики. — 1964 № 4(5). — с. 791–803.
4. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение — М.: ДМК Пресс, 2018. — 652 с.
5. Yurii Nesterov. A method of solving a convex programming problem with convergence rate O(1/k2). // Soviet Mathematics Doklady, 27(2):372–376, 1983.
6. Duchi J., Hazan E., Singer Y. Adaptive subgradient methods for online learning and stochastic optimization // Journal of Machine Learning Research, 2011, vol. 12, no. Jul. — P. 2121–2159
7. Tieleman T., Hinton G. Lecture 6.5 — RMSProp: Divide the Gradient by a Running Average of its Recent Magnitude, Coursera: Neural Networks for Machine Learning, 2012.
8. Kingma, D., Ba, J. Adam: A method for stochastic optimization. URL: https://arxiv.org/pdf/1412.6980.pdf (дата обращения: 07.02.2021)
9. Kaggle сайт. — URL: https://www.kaggle.com/c/dogs-vs-cats (дата обращения: 07.02.2021)
10. Шолле Франсуа, Глубокое обучение на Python. — СПб.: Питер, 2019. — 400 с.
11. Паттанаяк, Сантану, Глубокое обучение и TensorFlow для профессионалов. Математический подход к построению систем искусственого интеллекта на Python. — СПб.: «Диалектика», 2019. — 480 с.

© Крутов Тимофей Юрьевич (timofeykrutov@gmail.com),
Афанасьев Геннадий Иванович (gaipcs@bmstu.ru), Нестеров Юрий Григорьевич (ugn@bmstu.ru).
Журнал «Современная наука: актуальные проблемы теории и практики»


Московский государственный технический университет имени Н.Э. Баумана