УДК 004.032.26
DOI: 10.15587/2706-5448.2020.217613

ВЛИЯНИЕ МЕТОДА КОМПИЛЯЦИИ НА ОПРЕДЕЛЕНИЕ ТОЧНОСТИ ПОТЕРИ ОШИБКИ В ОБУЧЕНИИ НЕЙРОННОЙ СЕТИ

Аканова А., Калдарова М.

В области исследования NLP (Natural Language Processing) важным стало применение нейронной сети. Нейронная сеть широко применяется в семантическом анализе текстов на разных языках. В связи с актуализацией обработки больших данных на казахском языке была построенная нейронная сеть для проведения глубокого обучения. В данном исследовании объектом является процесс обучения глубокой нейронной сети, которая дает оценку алгоритму построения LDA модели. Одним из самых проблемных мест является определение правильных аргументов, которые при компилировании модели дадут оценку работы алгоритма. В ходе исследования использовался метод compile() из модульной библиотеки Keras, основными аргументами которого являются функция потерь, оптимизаторы, метрики. Нейронная сеть реализована на языке программирования Python. Основными аргументами компилятора глубокого обучения нейронной сети для оценки модели LDA является подбор аргументов для получения правильной оценки алгоритма построенной модели при помощи глубокого обучения нейронной сети. В качестве обучающихся данных представлен корпус текста на казахском языке с не более 8000 словами. С применением выше перечисленных методов был проведен эксперимент по выборке аргументов для компилятора модели при обучении корпуса текста на казахском языке. В результате оптимальными аргументами были выбраны оптимизатор – SGD, функция потерь – binary_crossentropy и метрика оценки – 'cosine_proximity', *которые в результате обучения показали стремление к 0 loss (ошибки)=0,1984, и cosine_proximity (точность обучения)=0.2239, что считается допустимыми мерами обучения. Полученные результаты показывают на правильный выбор аргументов компиляции. Данные аргументы можно применять при проведении глубокого обучения нейронной сети, где данными выборки выступает пара «тема и ключевые слова».*

Ключевые слова: *метрика оценки, качество обучения, алгоритмы оптимизации, энтропийная ошибка, нейронная сеть.*

1. Введение

Основным действием в глубоком обучении является корректировка весов для уменьшения ошибки с использованием серии обучающих примеров, которая в свою очередь сводится к поиску корреляции между входным и выходным слоями. Если корреляции нет в наличии, то ошибка никогда не достигнет 0 [1]. Обычно для проведения обучения создается нейронная сеть, с учетом параметров имеющихся данных, подвергающиеся обработке. Основной

фишкой глубокого обучения является использование оценки для корректировки значений весов с целью уменьшения потерь. Данная корректировка осуществляется оптимизатором, который реализует так называемый алгоритм обратного распространения ошибки: центральный алгоритм глубокого обучения [2]. Основными параметрами обучения обычно выступают: оптимизатор, коэффициент обучения, размер выборки, функция потерь, метрика оценки правильности построения маски, количество эпох обучения.

Поэтому актуальным является решение следующих вопросов:

– изучение библиотеки Keras;

– исследование применения функции потерь в глубоких обучениях нейронной сети;

– исследование алгоритмов оптимизации;

– исследование показателей качества.

Все выше перечисленные параметры будут реализованы в библиотеке Keras, и вручную не нужно будет производить вычислительный процесс. Но комбинация разных параметров может показать результаты, которые будут далеки от реальных показателей.

Таким образом, *объектом исследования* выбран процесс обучения глубокой нейронной сети, которая дает оценку алгоритму построения LDA модели. *Целью исследования* является выбор оптимальных параметров для определения точности потери ошибок при глубоком обучении нейронной сети.

2. Методика проведения исследования

В глубоком обучении нейронной сети вместе с точностью оптимизируется и сложность модели. Под сложностью понимается структурная сложность модели – это число параметров модели, с учетом их области определения. Альтернативой этому определению является статистическая сложность модели, то есть выдача минимального количества информации, которое требуется для передачи информации о модели и о выборке.

Оптимизатор один из трех параметров, используемых в компиляции глубокого обучения нейронной сети. Оптимизаторы или иначе алгоритмы оптимизации применяются при тренировке нейронной сети, которые служат для подбора весов таким образом, чтобы минимизировать ошибку на тренировочном наборе данных. В Keras реализованы такие основные алгоритмы оптимизации, как метод градиентного спуска, метод стохастического градиентного спуска – SGD, Adagrad, Adadelta, Adam (Adaptive Moment Estimation) и др. Оптимизатор – это величина, который показывает влияние отдельного признака на распределение целевой переменной. Для оптимизации процесса обучения был выбран оптимизатор SGD. Аргументом в данном оптимизаторе является learning_rate и уровень начального обучения должен быть float>=0. При применении SGD скорость обучения зависит от конкретных параметров, которые адаптированы к частому обновлению параметра во время обучения. Чем больше обновлений получает параметр, тем меньше скорость обучения. Другими словами, метод адаптивного градиента (SGD) [3] эффективно перераспределяет шаг обучения для

каждого отдельно взятого параметра, при этом учитываются все прошлые градиенты для данного параметра.

Коэффициент обучения подразумевает соотношения между обучающими и тестируемыми данными. Например, validation_split=0.25 означает, что 75 % данных будут применяться для обучения модели, а остальные 25 % будут использоваться для тестирования модели.

Размер выборки или batch_size определяет количество обучающих образцов, которое обрабатывается за одну итерацию алгоритма градиентного спуска.

Функция потерь или энтропия – это мера выдаваемой ошибки системой. Одной из самых популярных функций потерь, используемых в нейронных сетях, является кросс-энтропия (перекрестная энтропия), однако чаще в работе с NLP встречаются категорическая, разреженная или бинарная кросс-энтропия. Энтропия напрямую работает с неизвестным, это является важным аргументом в машинном обучении. Энтропия является функцией вероятности *p*. Чем больше вероятность события, тем меньше его неопределенность, то есть неизвестно событие произойдет или нет [4].

И третий не менее важный параметр – это метрика точности (accuracy). Метрики точности применяются в случае наличия недостоверности в выборке, то есть представители разных классов могут встречаться с разной вероятностью [5]. И для определения достоверности в Keras реализованы такие метрики точности, как accuracy, binary_accuracy, categorical_accuracy, sparse_categorical_accuracy, top_k_categorical_accuracy, cosine_proximity, sparse_top_k_categorical_accuracy, clone_metric. Точность определяет отношение правильно предсказанных объектов ко всем другим объектам. Обычно accuracy рассчитывают по формуле:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}, \tag{1}$$

где *TP* – True positive (истинно-положительное решение); *TN* – True negative (истинно-отрицательное решение); *FP* – False positive (ложно-положительное решение); *FN* – False negative (ложно-отрицательное решение).

В данном исследовании был применен binary_accuracy, который вычисляет частоту совпадения прогнозов с двоичными метками, в котором y_train должен совпадать или иметь параллельные данные с y_true. Для объективного тестирования классификатора в задачах с изображениями приемлемо использовать метрики ошибки MSE [6], однако для классификатора текстов больше подходит confusion matrix.

Для визуализации метрики точности применяется Матрица ошибок (confusion matrix), которая обычно представляется, как на рис. 1.

Каждая позиция вычисляется относительно формул (2)–(5), где *n* – это количество классов, метка класса принимает значение +1 (положительный класс) или –1 (отрицательный класс). Вводятся 4 значения, соответственно:

$$TP\ (True\ positive) = \sum_{i=0}^{n} \left[a\left(x_i\right) = +1 \right] \left[y_i = +1 \right], \tag{2}$$

$$TN\ (True\ negative) = \sum_{i=0}^{n}\left[a(x_i) = -1\right]\left[y_i = -1\right], \quad (3)$$

$$FP\ (False\ positive) = \sum_{i=0}^{n}\left[a(x_i) = +1\right]\left[y_i = -1\right], \quad (4)$$

$$FN\ (False\ negative) = \sum_{i=0}^{n}\left[a(x_i) = -1\right]\left[y_i = +1\right], \quad (5)$$

где $a(x_i)$ – класс с элементами; y_i – результат алгоритма обучения

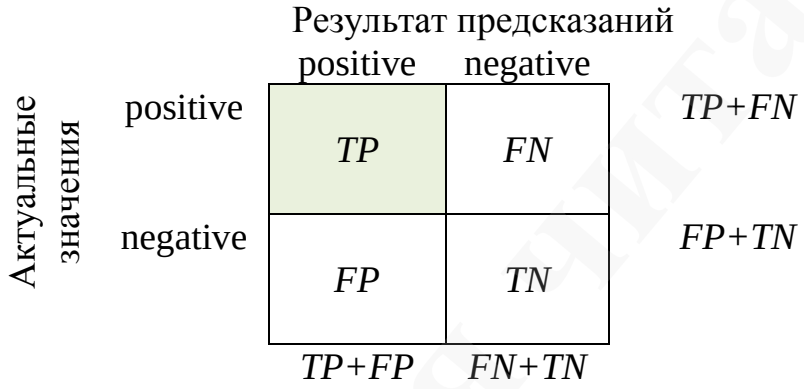


Рис. 1. Матрица ошибок

Таким образом, раздел *TP* показывает сумму количества классов $a(x_i)$, в котором пара «тема и ключевые слова» имеют высокую точность и являются истинными и по предсказанию, и по результатам y_i алгоритма обучения (2). Раздел *TN* показывает сумму количества классов $a(x_i)$, в котором пара «тема и ключевые слова» имеет ложную точность, и по предсказанию, и по результатам y_i алгоритма обучения (3). Раздел *FP* показывает сумму количества классов $a(x_i)$, в котором пара «тема и ключевые слова» имеет истинную точность по предсказанию, и ложную по результатам y_i алгоритма обучения (4). Раздел *FN* показывает сумму количества классов $a(x_i)$, в котором пара «тема и ключевые слова» имеет ложную точность по предсказанию, и истинную точность по результатам y_i алгоритма обучения (5). В разделе *FN* и *FP* система показывает ошибки, которые были получены в ходе обучения. В экспериментах должно учитываться определенное количество данных. Если имеет место увеличение порога данных, то выдается меньше ложно-положительных ошибок и больше ложно-отрицательных ошибок. Поэтому одна из кривых может расти, а вторая – падать. По такому графику можно подобрать оптимальное значение порога. Если такого порога не нашлось, нужно обучать другой алгоритм.

3. Результаты исследования и их обсуждение

На момент данного исследования имеется нейронная сеть из последовательности слоев: Embedding(), SpatialDropout1D(), LSTM(dropout,

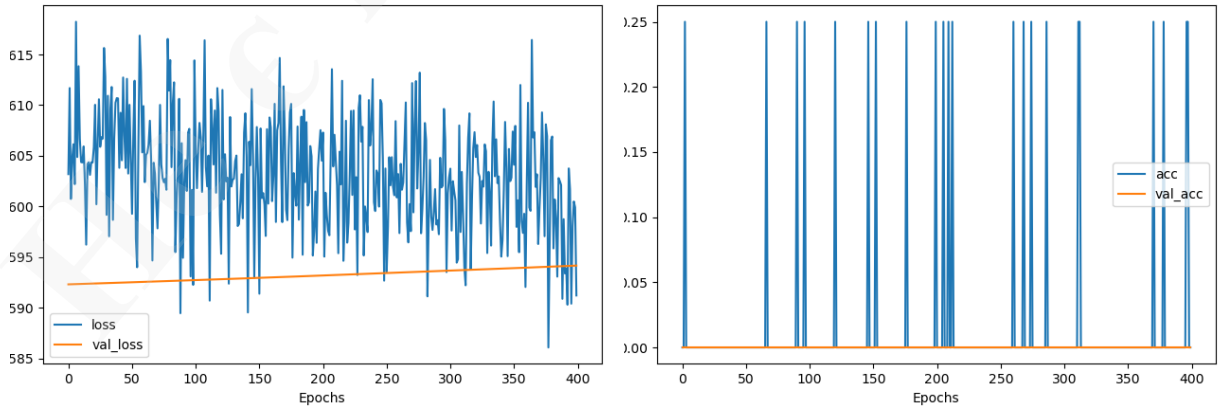
recurrent_dropout), Dense()), которая представляет собой традиционную многослойную сеть с прямой связью, использующие обратное распространение ошибок [7]. На предыдущих исследованиях были определены параметры регуляризации в слоях нейронной сети, которые составляют 70 % данных для слоя SpatialDropout1D, для скрытых слоев dropout и recurrent_dropout. В результате проведенных экспериментов была получена комбинация оптимальных параметров в компиляции модели нейронной сети. Приведем несколько примеров из проведенных экспериментов [8]:

1-ая комбинация. model.compile(loss='sparse_categorical_crossentropy', optimizer='adam'), результат которого изображен на рис. 2. Данная комбинация показывает, что в shape меняется количество классов на 1, дальше процесс останавливается и показывает ошибку.

```
  history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,validation_split=0.2)#запись результато
File "C:\Users\Simple\Anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py", line 1263, in fit
  validation_split=validation_split)
File "C:\Users\Simple\Anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py", line 907, in _standar
_user_data
  exception_prefix='target')
File "C:\Users\Simple\Anaconda3\lib\site-packages\tensorflow\python\keras\engine\training_utils.py", line 191, in st
rdize_input_data
  ' but got array with shape ' + str(data_shape))
lueError: Error when checking target: expected dense to have shape (1,) but got array with shape (19,)
```

Рис. 2.Результат применения 1-ой комбинации параметров

2-ая комбинация. model.compile(loss='poisson', optimizer='Adadelta', metrics =['accuracy']) – данная комбинация показывает стремление к 0.5 результатов вычисления ошибок в обучающей и стремление к 0 тренировочной модели [9]. И в точности обучения ни тренировочная, ни обучающая модели не меняется в течение обучения. Это не соответствует результатам вычисления, производимыми accuracy (рис. 3).



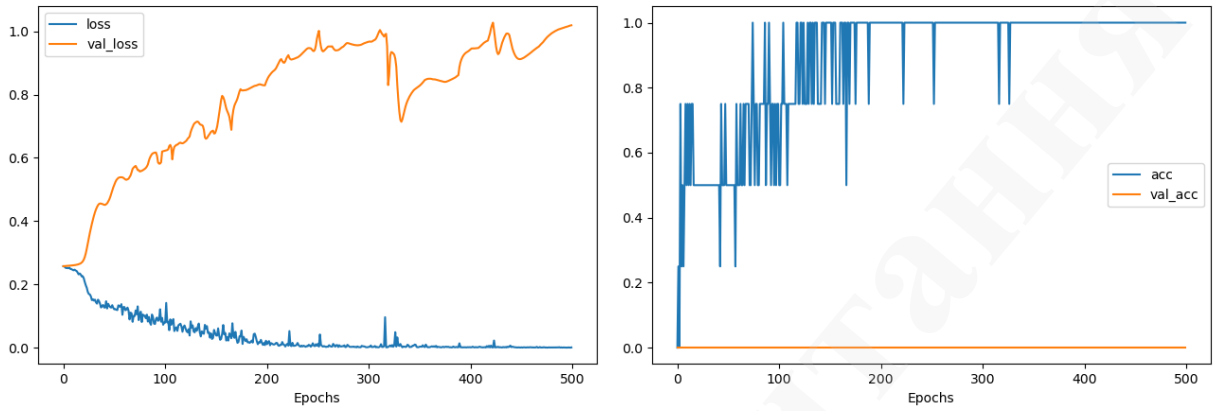
а *б*

Рис. 3. Результат 2-ой комбинации:
а – ошибка (loss, val_los); *б* – точность(acc, val_acc)

3-я комбинация. model.compile (loss='poisson', optimizer='adam', metrics=['accuracy']) – данная комбинация параметров показывает, что функция ошибки расходится после *n*-ой итерации. Это говорит о необходимости

переобучения с места расхождения обучающихся и тестируемых данных, точность в свою очередь не меняется на протяжении обучения.

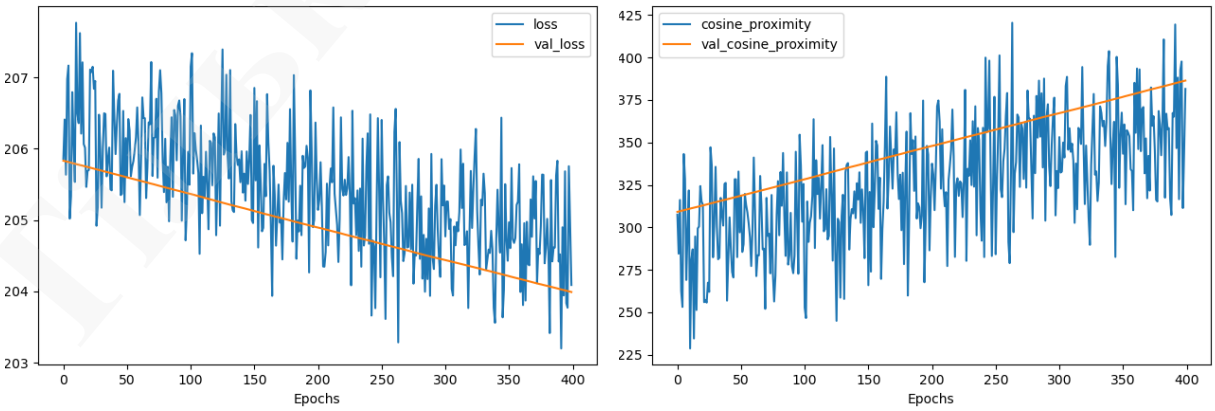
Значит, результат является недостоверным в данном случае (рис. 4).



а *б*

Рис. 4. Результат 3-ей комбинации:
а – ошибка (loss, val_los); *б* – точность (acc, val_acc)

4-ая комбинация. model.compile(loss='binary_crossentropy', optimizer='SGD', metrics=['cosine_proximity']) – данная комбинация параметров показывает, что ошибка в обучающихся данных приближается к 0 параллельно. Это означает что обучение проходит нормально без переобучения. Точность в расчете потери ошибки, соответственно, увеличивается и стремится к единице. Результатом loss (ошибки обучения)=0.1984, cosine_proximity (точность ошибки)=0.2239, val_loss (ошибка тестирования)=0.1985, val_cosine_proximity (точность ошибки тестирования)=0.2235 (рис. 5).



а *б*

Рис. 5. Результат 4-ой комбинации:
а – ошибка (loss, val_los); *б* – точность(acc, val_acc)

Это и требовалось получить в результате эксперимента.

4. Выводы

Таким образом, имеется последовательная композиция нейронной сети: Embedding(), SpatialDropout1D(0.7)), LSTM(64, dropout=0.7, recurrent_ dropout=0.7)), Dense()) с предварительными параметрами emb_dim=128, n_most_common_words=8000, batch_size=256, epochs=500. В результате проведенных экспериментов с использованием различных комбинаций аргументов в методе компиляции модели был выявлен оптимальный вариант. Оптимальной комбинацией аргументов компилятора при определении точности потери ошибки являются параметры: binary_crossentropy, optimizer=SGD, metrics=cosine_proximity. Результаты исследования применяются при проведении глубокого обучения нейронной сети, где выборкой являются текстовые данные. Аргументы используются в методе compile() библиотеки Keras для языка программирования Python [10].

Литература

1. Trask, E. (2020). *Glubokoe obuchenie*. Saint Petersburg: Piter, 352.
2. Sholle, F. (2018). *Glubokoe obuchenie na Python.* Saint Petersburg: Piter, 400.
3. Duchi, J., Hazan E., Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, *12*, 2121–2159.
4. Kononiuk, A. E. (2011). *Informatsiologiia. Obschaia teoriia informatsii. Kniga 3*. Kyiv: Osvita Ukraini, 412.
5. Dzhulli, A., Pal, S. (2018). *Biblioteka Keras – instrument glubokogo obucheniia*. Moscow: DMK Press, 294.
6. Koyuncu, H. (2020). Loss Function Selection in NN based Classifiers: Try-outs with a Novel Method. *2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. doi: http://doi.org/10.1109/ecai50035.2020.9223208
7. Hung, C.-C., Song, E., Lan, Y. (2019). Foundation of Deep Machine Learning in Neural Networks. *Image Texture Analysis*, 201–232. doi: http://doi.org/10.1007/978-3-030-13773-1_9
8. *Metriki*. Available at: https://ru-keras.com/metric/
9. Ketkar, N. (2017) Introduction to Keras. *Deep Learning with Python.* Berkeley: Apress. doi: http://doi.org/10.1007/978-1-4842-2766-4_7
10. Chollet, F. (2017). *Deep Learning With Python.* Black & White, 384. Available at: https://github.com/fchollet/deep-learning-with-python-notebooks