****

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Институт информатики и кибернетики

Кафедра информационных систем и технологий

Отчет по лабораторной работе №3

по курсу

«Интеллектуальные системы управления»

Выполнил: студент группы

6231

Иванов И.И.

Проверил:

к.т.н., доцент Солдатова О.П.

Самара 2023

Задание №6

Цель работы

Целью данной лабораторной работы является демонстрация способности нейронной сети решать задачи классификации. Реализовать модель на языке Python в среде Jupyter Notebook с использованием библиотек Numpy, Matplotlib, Keras и Tensorflow.

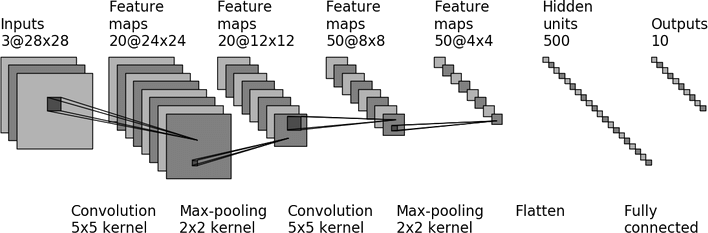
Используемая модель: простая свёрточная сеть (до 10 слоёв).

Порядок выполнения работы. Данная лабораторная работа выполняется в несколько этапов:

1. Выбрана предметная область изображения собак и кошек.
2. Написать программу, имитирующую работу свёрточной сети (в составе должны быть слои свёртки, пулинга, и полносвязные слои) и провести обучение сети с использованием алгоритмов Adam , Adagrad , Adadelta, RMSProp , Nadam .
3. Провести тестирование сети, обученной при помощи всех алгоритмов.
4. Продемонстрировать работу свёрточной сети преподавателю.
5. Исследовать зависимость функции погрешности при обучении и классификации, а также метрику качяества accuracy от объёма обучающей выборки для всех алгоритмов.
6. Исследовать зависимость функции погрешности при обучении и классификации, а также метрику качества accuracy от числа эпох обучения для всех алгоритмов.
7. Исследовать зависимость функции погрешности при обучении и классификации, а также метрику качества accuracy от размера пакета (мини батча) обучения для всех алгоритмов.
8. Построить графики функции погрешности при обучении и классификации, а также метрики качества accuracy для всех исследований.
9. Составить отчёт, который должен содержать постановку задачи, обучающие выборки, результаты исследований по пунктам 5-8 и сдать его преподавателю.

ОПИСАНИЕ

Свёрточная нейронная сеть — специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 год и нацеленная на эффективное распознавание образов, входит в состав технологий глубокого обучения. Архитектура свёрточной нейронной сети представлена на рисунке 1.

  
Рисунок 1 – Архитектура свёрточной нейронной сети

В качестве предметной области был выбран набор данных из 8 000 изображений с 2 категориями: собаки, кошки.

Тестовый набор содержит 2000 изображений. Пример изображения из обучающей выборки представлен на рисунке 2, пример из тестового набора представлен на рисунке 3.



Рисунок 2 – Пример изображения из обучающей выборки

Обучение выполняется по следующему алгоритму:

1. Цветные изображения преобразовываются к размерам 64х64;
2. Первый слой и второй слой сети выполняет свертку входного изображения, в результате чего получается слой размером 60х60х32;
3. Полученный слой отбирается с помощью слоя max-pool с размером фильтра 2 и шагом 2, в результате чего получается слой размером 30х30х32;
4. Далее выполняем свёртку с размером фильтра 3 (четвертый и пятый слои);
5. Полученный слой отбирается с помощью слоя max-pool с размером фильтра 2 и шагом 2;
6. Далее усредняем все значения по последней оси на слое 7;
7. Объединяем с размером фильтра 2 и шагом 2;



Рисунок 3 – Пример изображения из тестовой выборки

Построение модели

В составе модели есть слои свёртки (Conv2D(32, (3,3),…), Conv2D(64, (3,3),…) в качестве функции активации использовалась функция Relu), слои пулинга после каждого сверточного слоя (MaxPooling2D((2, 2)). MaxPooling2D — это способ уменьшить количество параметров модели. GlobalAveragePooling2D()) MaxPooling2D —способ усреднения значений по последней оси. Завершают архитектуру модели полносвязный слой: Dense(units=2, activation='softmax'). Конечный слой имеет выходной размер 2, соответствующий 2 классам изображений. Вся архитектура модели представлена ниже:

*# Initalize CNN*classifier = Sequential()  
  
*# Add 2 convolution layers*classifier.add(Conv2D(filters=32, kernel\_size=(3,3), input\_shape=(64, 64, 3), activation='relu'))  
classifier.add(Conv2D(filters=32, kernel\_size=(3,3), activation='relu'))  
  
*# Add pooling layer*classifier.add(MaxPooling2D(pool\_size=(2,2)))  
  
*# Add 2 more convolution layers*classifier.add(Conv2D(filters=32, kernel\_size=(3,3), activation='relu'))  
classifier.add(Conv2D(filters=32, kernel\_size=(3,3), activation='relu'))  
  
*# Add max pooling layer*classifier.add(MaxPooling2D(pool\_size=(2,2)))  
  
*# Add 2 more convolution layers*classifier.add(Conv2D(filters=32, kernel\_size=(3,3), activation='relu'))  
classifier.add(Conv2D(filters=32, kernel\_size=(3,3), activation='relu'))  
  
*# Add max pooling layer*classifier.add(MaxPooling2D(pool\_size=(2,2)))  
  
*# Add global average pooling layer*classifier.add(GlobalAveragePooling2D())  
  
classifier.add(Dropout(0.5))  
  
*# Add full connection*classifier.add(Dense(units=2, activation='softmax'))

После построении модели нужно ее скомпилировать и затем можно обучать ее. Когда мы компилируем модель, мы объявляем функцию потерь и оптимизатор. В качестве оптимизатора для первоначального исследования модели примем Adam, а так, как в этой работе стоит задача классификации данных, предсказывая вероятность того, принадлежат ли данные к одному или другому классу, в качестве функции потерь возьмем функцию кросс-энтропийных потерь:

*# Compiling the ANN*classifier.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['acc'])

Для обучения размер пакета (мини батча) был выбран равным 32. Сам метод обучения fit представлен ниже:

classifier.fit(  
train\_set,

steps\_per\_epoch=len(train\_set),epochs=50,  
validation\_data=test\_set,  
callbacks = callback\_list,  
validation\_steps=10)

Результат обучения представлен на рисунках ниже (зависимости погрешности при обучении на рис.1 и зависимость метрики качества accuracy при обучении от числа эпох на рис.2). Из рисунка 2 видно, что после 30 эпохи происходит переобучение модели.

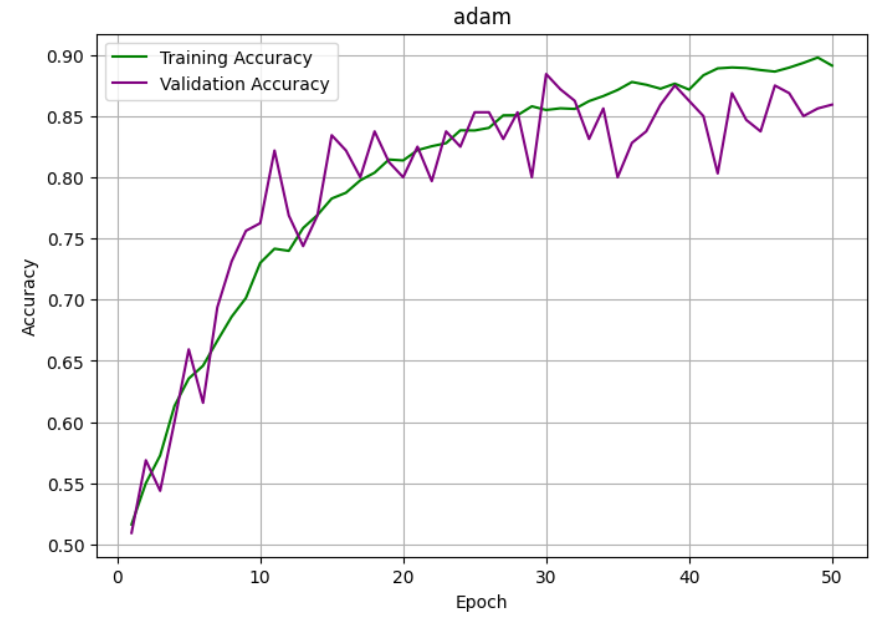


Рисунок 4 – Зависимость погрешности обучения от числа эпох для алгоритма Adam

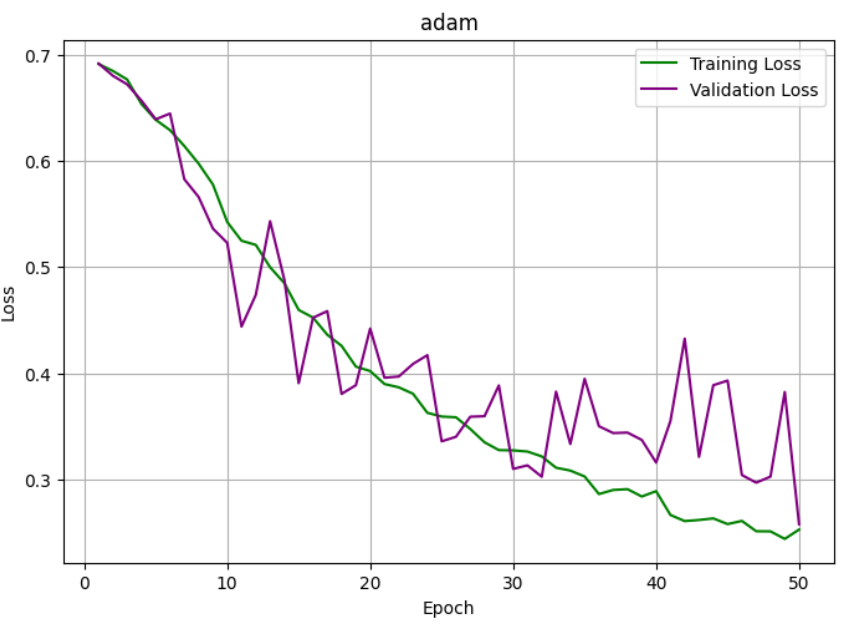


Рисунок 5 – Зависимость метрики качества accuracy при обучении от числа эпох для алгоритма Adam

Таблица 1. Результаты изменения погрешности классификации и метрики accuracy на тестовом наборе

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 5 эпох | 10 эпох | 30 эпох | 50 эпох |
| Loss | 0.64 | 0.52 | 0.31 | 0.26 |
| Accuracy | 0.66 | 0.76 | 0.88 | 0.86 |

Исследования

1. Исследовать зависимость функции погрешности при обучении и классификации, а также метрику качества accuracy от объёма обучающей выборки для всех алгоритмов.

Для этого исследования примем:

- размер пакета (мини батча) = 32;

- количество эпох = 1;

- функция погрешности - categorical\_crossentropy.

На рисунке 6 показана зависимость функции погрешности при обучении от объёма обучающей выборки для всех алгоритмов.

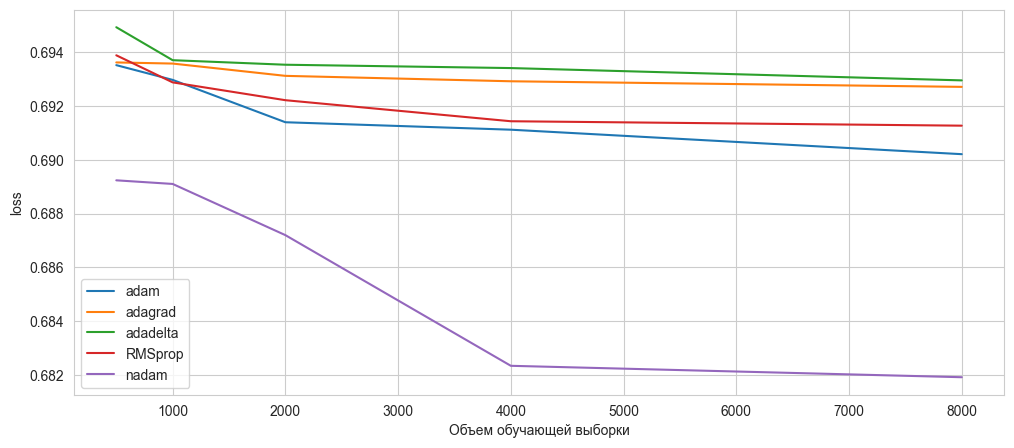


Рисунок 6 – Зависимость погрешности при обучении от объёма обучающей выборки для всех алгоритмов

Как видно из рис.6 наименьшей погрешностью при обучении обладают алгоритмы Adam, Nadam и RMSprop, но наилучшие показатели принадлежат алгоритму Nadam.

На рисунке 7 показана зависимость метрики качества accuracy при обучении от объёма обучающей выборки для всех алгоритмов.

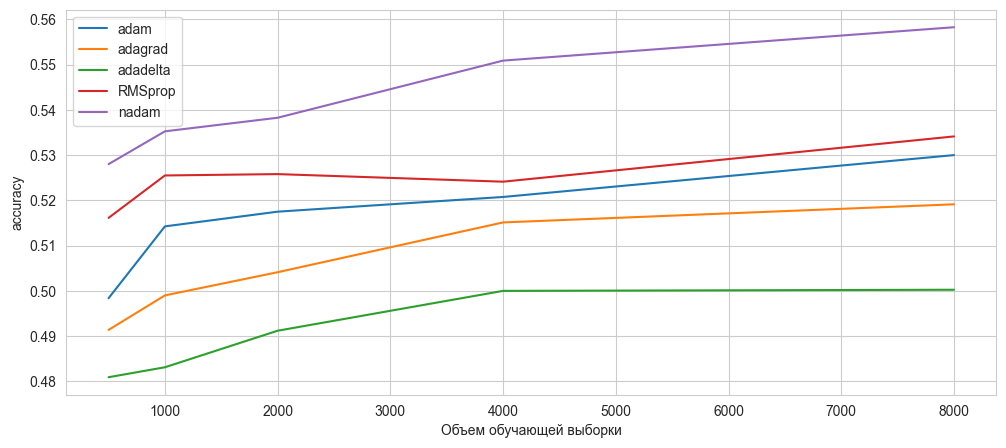


Рисунок 7 – Зависимость accuracy при обучении от объёма обучающей выборки для всех алгоритмов

Как видно из рис.7 наилучшей метрикой accuracy при обучении обладают алгоритмы Adam, Nadam и RMSprop, наилучшие показатели принадлежат алгоритму Nadam.

На тестовом наборе наблюдаются аналогичные результаты при изменении объёма обучающей выборки, как и на обучающем наборе (рисунок 8, рисунок 9).

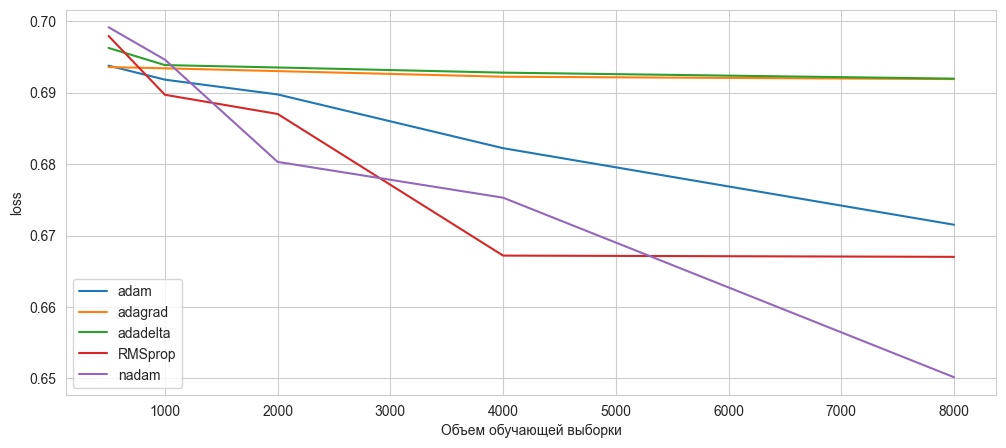


Рисунок 8 – Зависимость погрешности при тестировании от объёма обучающей выборки для всех алгоритмов

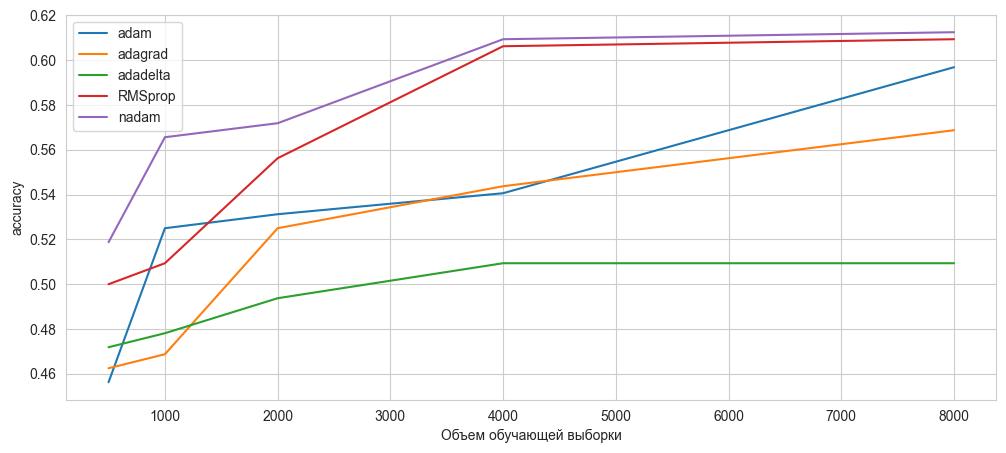


Рисунок 9 – Зависимость accuracy при тестировании от объёма обучающей выборки для всех алгоритмов

Можно сделать вывод, что функция погрешности при обучении и классификации, а также метрика качества accuracy в большинстве случаев обладают наилучшими результатами при наибольшем объёме обучающей выборки.

1. Исследовать зависимость функции погрешности при обучении и классификации, а также метрику качества accuracy от числа эпох обучения для всех алгоритмов.

Для этого исследования примем:

- размер пакета (мини батча) = 32;

- 8000 изображений в обучающем и 2000 в тестовом наборе;

- функция погрешности - categorical\_crossentropy;

На рисунках 7 и 8 показана зависимость функции погрешности при обучении от числа эпох обучения для всех алгоритмов. После 30 эпох все алгоритмы показывают переобучение модели. При увеличении количества эпох уменьшается погрешность для всех алгоритмов.

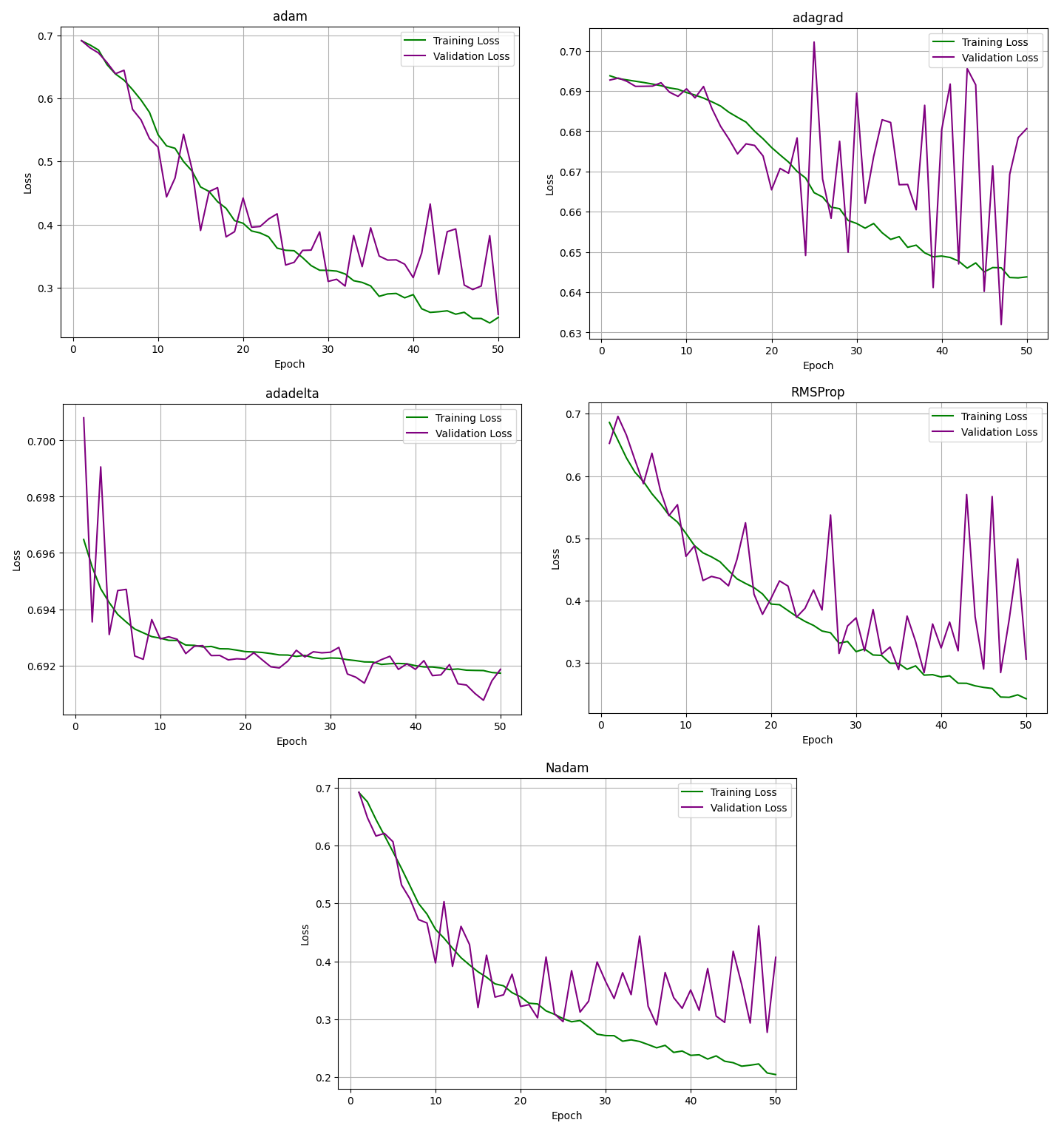


Рисунок 10 – Зависимость погрешности при обучении от числа эпох для всех алгоритмов

При увеличении числа эпох accuracy увеличивается для всех алгоритмов.

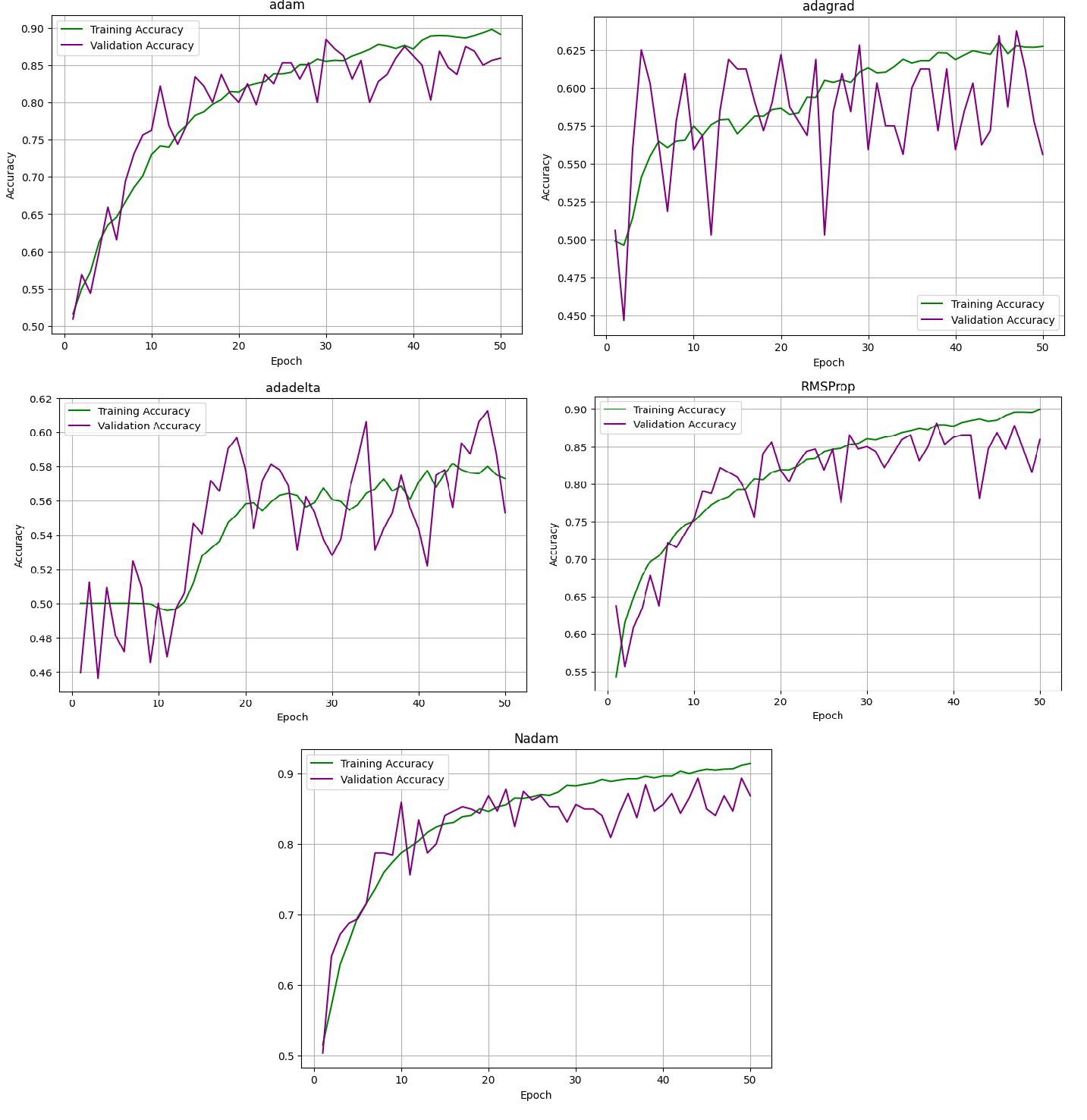


Рисунок 11 – Зависимость метрики качества accuracy от числа эпох для всех алгоритмов

Таблица 3. Результаты изменения погрешности классификации и метрики accuracy на тестовом наборе от алгоритма обучения за 30 эпох обучения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Adam | Adagrad | Adadelta | RMSprop | Nadam |
| Loss | 0.3097 | 0.6895 | 0.6925 | 0.3722 | 0.3653 |
| Accuracy | 0.8844 | 0.5594 | 0.5281 | 0.8500 | 0.8562 |

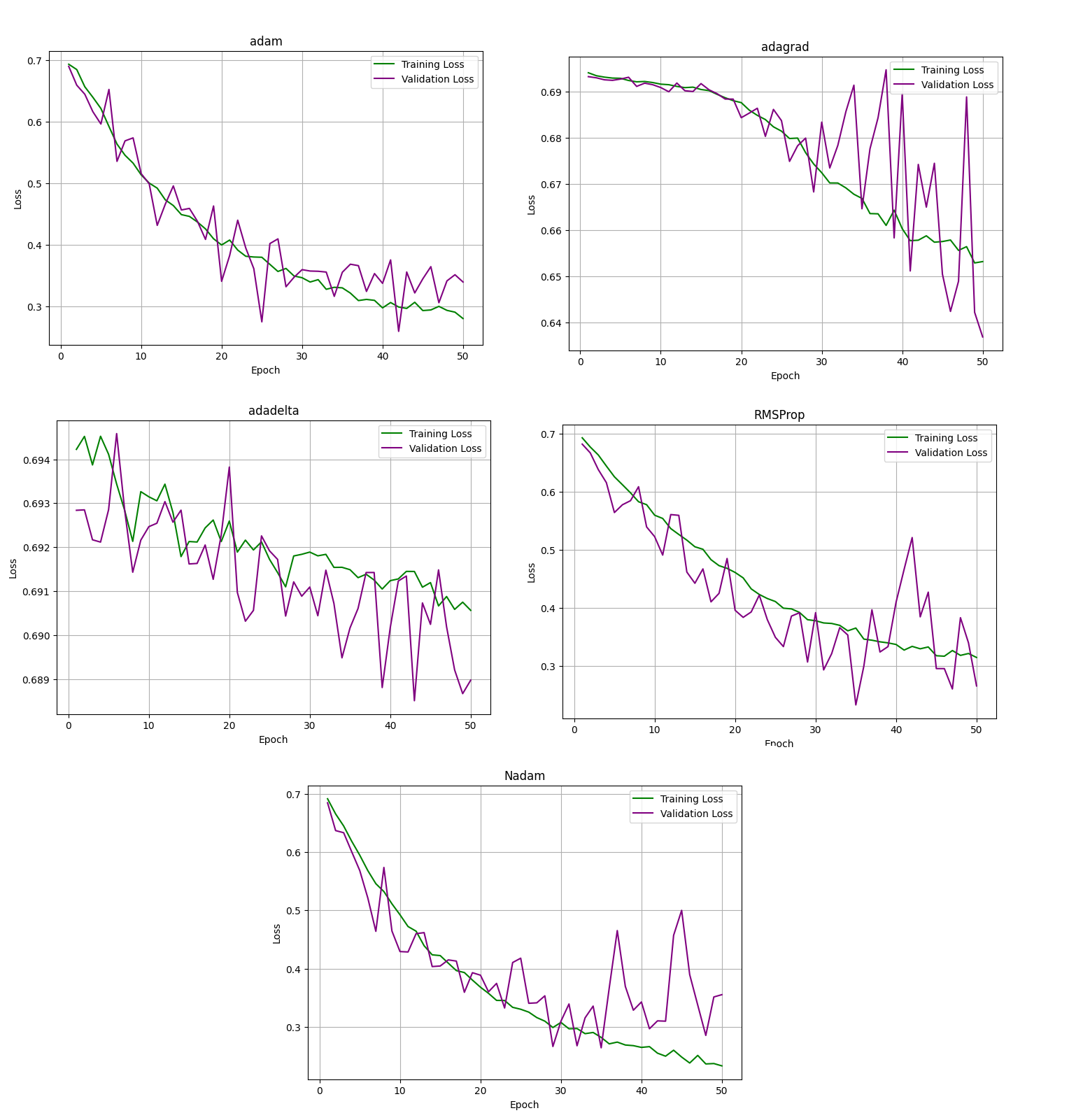


Рисунок 12 – Зависимость погрешности при обучении от числа эпох для всех алгоритмов, где dropout равен 0,5

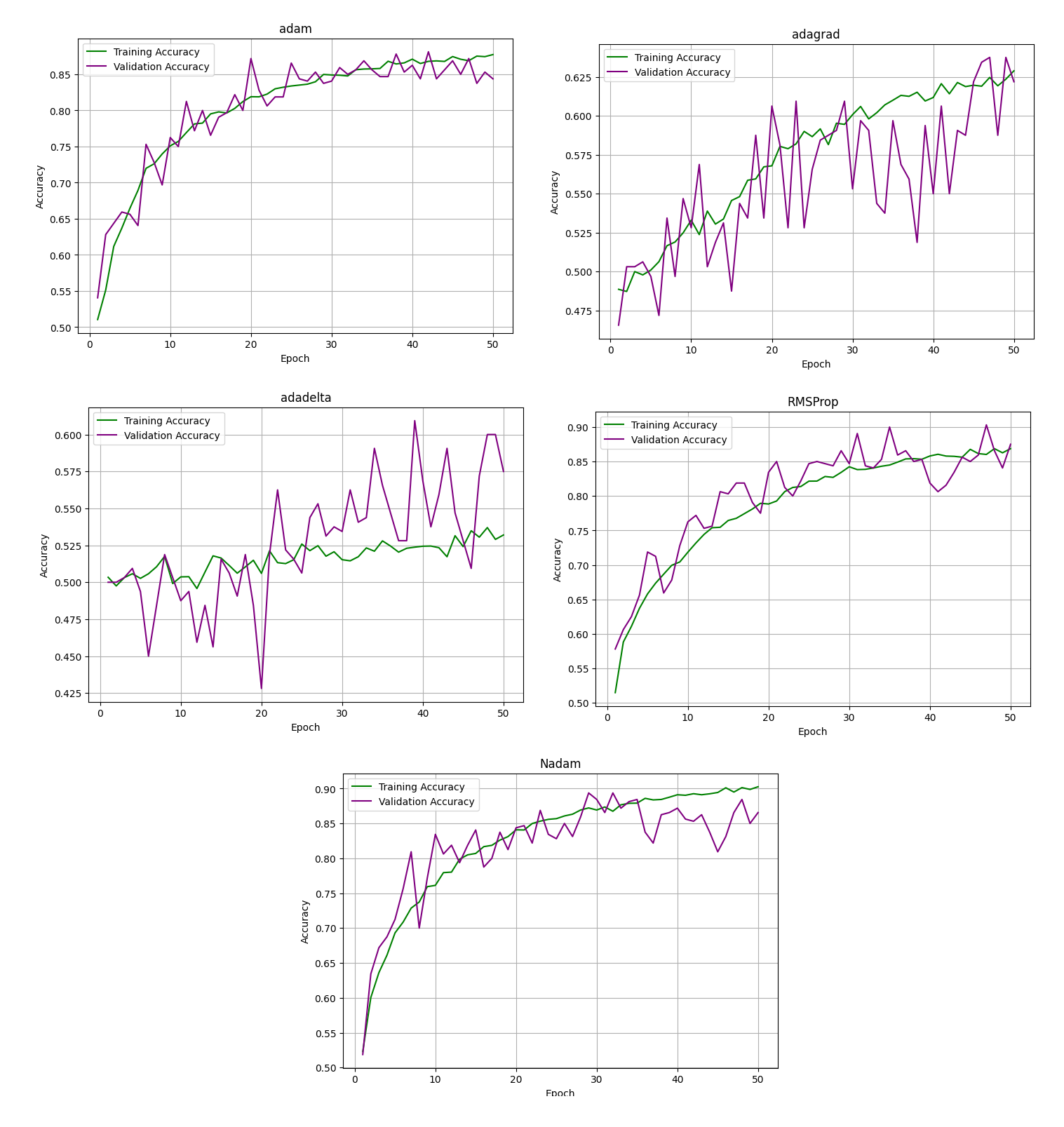


Рисунок 13 – Зависимость метрики качества accuracy от числа эпох для всех алгоритмов, где dropout равен 0,5

Самые лучшие результаты этого исследования показали алгоритмы Adam, Nadam и RMSprop.Оптимальное значение dropout для всех алгоритмов является 0,5.

1. Исследовать зависимость функции погрешности при обучении и классификации, а также метрику качества accuracy от размера пакета (мини батча) обучения для всех алгоритмов.

Для этого исследования примем:

- размер пакета (мини батча) = 32;

- количество эпох = 1;

- функция погрешности - categorical\_crossentropy.

На рисунках 14 и 15 показаны зависимости функции погрешности при обучении и метрики качества accuracy при обучении от размера пакета (мини батча) обучения для всех алгоритмов соответственно. При увеличении размера пакета (мини батча) функция погрешности при обучении увеличивается для всех алгоритмов, а accuracy в общем случае уменьшается.

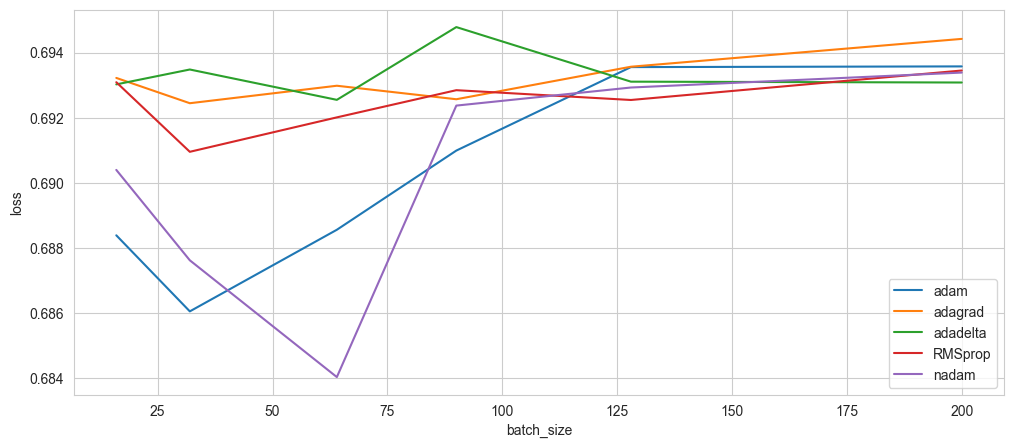


Рисунок 14 – Зависимость погрешности обучения от размера пакета (мини батча) обучения для всех алгоритмов

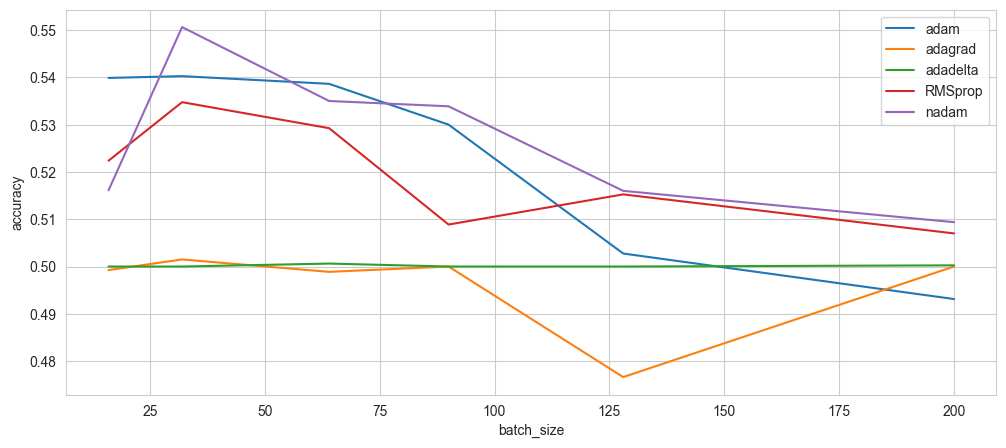


Рисунок 15 – Зависимость метрики качества accuracy от размера пакета (мини батча) обучения для всех алгоритмов

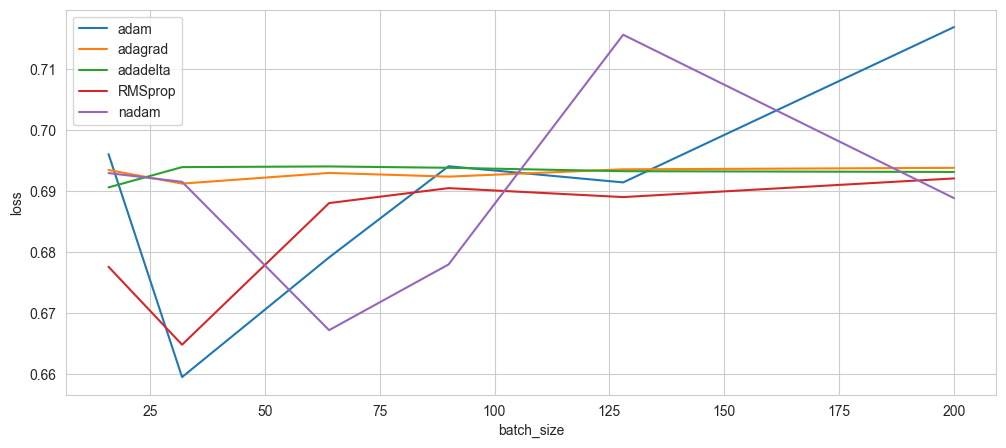


Рисунок 16 – Зависимость погрешности тестирования от размера пакета (мини батча) обучения для всех алгоритмов

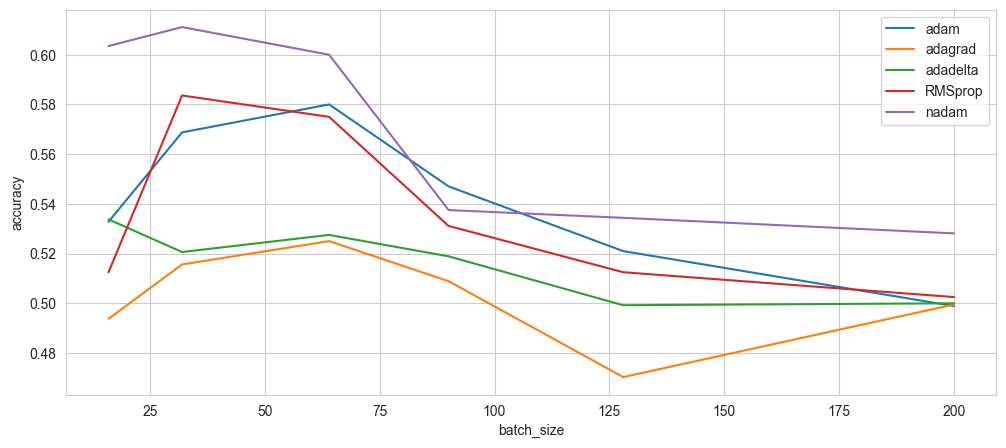


Рисунок 17– Зависимость метрики качества accuracy от размера пакета (мини батча) обучения для всех алгоритмов

Так же, как и в предыдущем исследовании наилучшие показатели погрешности и accuracy принадлежат алгоритмам Nadam и RMSprop при размере пакета равным 32.

Выводы

В результате работы была построена простая свёрточная сеть на языке программирования Python и проведено обучение сети с использованием библиотек Keras и Tensorflow. Были проведены различные исследования данного алгоритма обучения с применением библиотек Numpy и Matplotlib для построения графиков.

Исходя из проведенных исследований, наилучшие результаты по прогнозу дают для алгоритма

Adam:

- количество эпох = 20;

- 8000 изображений в обучающем, 2000 в тестовом наборе;

- размер пакета (мини батча) = 32.

Adagrad:

- количество эпох = 20;

- 8000 изображений в обучающем, 2000 в тестовом наборе;

- размер пакета (мини батча) = 32.

Adadelta:

- количество эпох = 10;

- 8000 изображений в обучающем, 2000 в тестовом наборе;

- размер пакета (мини батча) = 32.

RMSprop:

- количество эпох = 15;

- 8000 изображений в обучающем, 2000 в тестовом наборе;

- размер пакета (мини батча) = 32.

Nadam:

- количество эпох = 20;

- 8000 изображений в обучающем, 2000 в тестовом наборе;

- размер пакета (мини батча) = 64.

Также в результате исследований были отмечены алгоритмы Adam, Nadam и RMSprop, как наилучшие в обучении.