САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Сабреков Артём Азатович

Выпускная квалификационная работа

Применение искусственных нейронных сетей в задачах распознавания изображений

Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2016 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Процессы управления и высокопроизводительные вычислительные системы»

Научный руководитель:
доцент кафедры теории систем управления электрофизической аппаратурой, к. ф. - м. н.,
Козынченко Владимир Александрович.

Рецензент:
доцент кафедры компьютерного моделирования и многопроцессорных систем, к. ф. - м. н.,
Гришкин Валерий Михайлович.

Санкт-Петербург

2020 г.

Содержание

Введение.

Распознавание изображений имеет множество сфер применения, таких как видеонаблюдение (распознавание лиц), машинный перевод (перевод текстов с одного языка на другой), маркетинг (анализ изображений товаров), медицина (распознавание заболеваний на снимках), беспилотные автомобили (распознавание объектов на видео), поисковые системы и многие другие.

В настоящее время для распознавания изображений преимущественно используются искусственные нейронные сети. Они представляют из себя некоторую программно-реализуемую вычислительную модель, которая способна обрабатывать данные и на их основе обучаться решать поставленную ей задачу.

В 2010 году начался проект ImageNet Large Scale Visual Recognition Challenge (ILSVRC). В рамках которого разработчики ежегодно соревнуются в классификации изображений базы данных ImageNet. По состоянию на август 2017 года ImageNet состоит из более чем 14 миллионов изображений, разбитых на более чем 21 тысячу классов. Если в 2011 году лучшая верность предсказаний для задачи распознавания ImageNet составляла всего 50.9%, то на данный момент она составляет уже 88.5%. Это демонстрирует стремительные темпы развития в данной области.

Выпускная квалификационная работа состоит из введения, трёх глав, заключения и списка использованной литературы.

В первой главе приведены некоторые теоретические сведения, подводящие к понимаю искусственных нейронных сетей. В ней рассмотрены основные понятия, которые использованы при описании моделей распознавания.

Во второй главе раскрыта постановка задачи распознавания изображений. В ней также приведена информация про набор данных, его предобработку, а также некоторая информация про гиперпараметры обучения и метрики.

Третья глава посвящена вычислительному эксперименту. В этой главе описываются полученные результаты и проводится их анализ. Формулируются выводы.

В заключении подводятся итоги проделанной работы.

Цель и задачи.

Данная работа посвящена решению задачи распознавания изображений с использованием искусственных нейронных сетей. Для распознавания был выбран набор изображений CIFAR-10, состоящий из 60 000 цветных изображений, принадлежащих 10-ти непересекающимся классам. Целью работы будет являться реализация нейронных сетей, способных, обучившись на одной части изображений, правильно классифицировать вторую часть набора.

Задачи:

1. Предобработка изображений набора.
2. Реализация нескольких архитектур искусственных нейронных сетей
3. Обучение сетей распознаванию набора изображений CIFAR-10.
4. Анализ и сравнение полученных результатов.

Обзор литературы.

Задача распознавания изображений впервые возникла еще в середине XX-го века, однако первые рабочие модели нейронных сетей появились лишь в 1990-1991 годах. Активное же развитие нейронных сетей началось с 2010-2012 годов, это связано с увеличением мощности GPU видеокарт и началом вышеупомянутого проекта ImageNet, в котором в 2012 году нейросеть показала лучший результат. С того времени по данной теме было написано огромное количество книг и статей. Среди них есть те, что углубляются в математику процесса, как например Гудфеллоу Я., Бенджио И., Курвилль А., «Глубокое обучение», в которой помимо самих искусственных нейронных сетей можно увидеть необходимую для их понимания математическую базу. Также есть и издания более прикладного характера, как например книга Антонио Джулли, Суджит Пал. «Библиотека Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow», в которой акцент ставится больше на программной реализации нейронных сетей.

Отдельно хочется отметить книгу Орельена Жерона «Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем». В ней доступным языком довольно подробно описываются многие моменты как нейронных сетей, так и машинного обучения в целом. При этом показаны способы реализации описанных моделей при помощи библиотек Scikit-Learn и TensorFlow.

Глава 1. Искусственные нейронные сети.

1.1. Модель искусственного нейрона.

Большая часть искусственных нейронных сетей является комбинацией некоторого набора структурных элементов, которые называют слоями нейронных сетей. Слои же в свою очередь состоят из нейронов. Существует множество моделей нейронов, опишем самую часто используемую. Такой нейрон представляет из себя некоторый вычислительный элемент, у которого есть множество входов и один выход. На каждый вход поступают вещественные числа, каждое из которых умножается на некоторый коэффициент, называемый весом. Далее, получившиеся числа суммируются, также ним прибавляется некоторая константа. Выходом нейрона будет являться значение, полученное в результате применения к нашему итоговому числу некоторой функции. Данную функцию называют функцией активации. Таким образом, нейрон можно рассматривать как функцию от линейной комбинации входных значений.

$$Y = f(\sum_{i=1}^{n} X_i \times W_i + W_0),$$

где Y – выход нейрона, X_i – i-ая компонента входа, W_i – i-ый вес, W_0 – константа, обычно равная единице.

Рассмотрим некоторые наиболее часто используемые функции активации.

На рисунках 1.1 – 1.5 изображены графики данных функций.

1. Ступенчатая функция активации.

   $$f(x) = \{0, x \leq 0; 1, x > 0\}$$

2. Линейная функция активации

   $$f(x) = cx + b$$

3. Сигмоидальная функция активации

   $$f(x) = \frac{1}{1+e^{-x}}$$

4. Гиперболический тангенс

   $$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

5. ReLU

$$f(x) = \max(0, x)$$

6. Softmax

$$\sigma(x)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

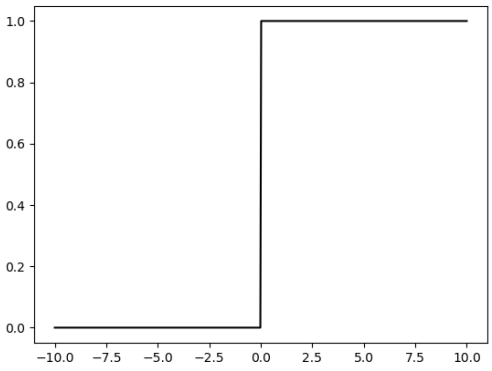

Рис. 1.1. Ступенчатая функция

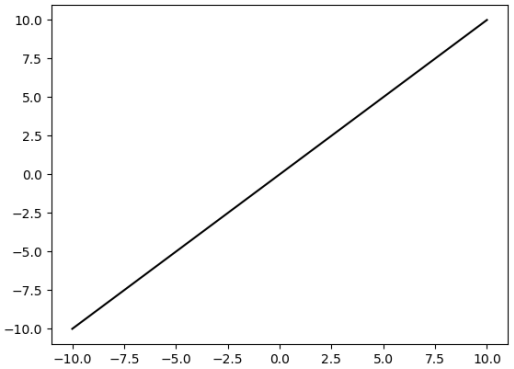

Рис. 1.2. Линейная функция

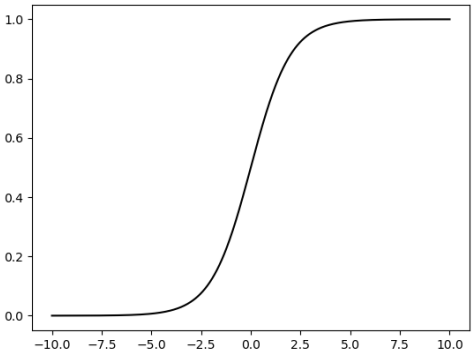

Рис. 1.3. Сигмоидальная функция

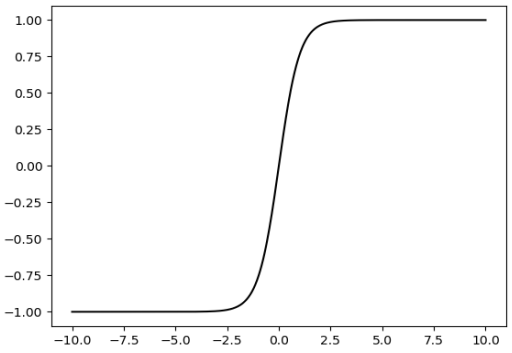

Рис. 1.4. Гиперболический тангенс

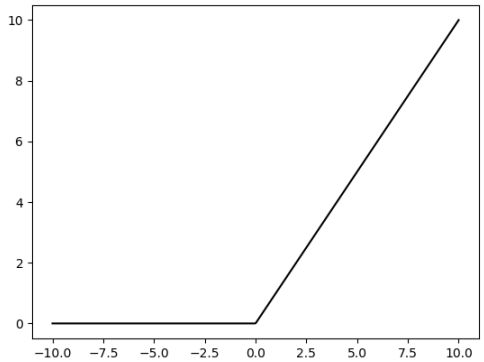

Рис. 1.5. ReLU

1.2. Сети прямого распространения.

Для распознавания изображений обычно используются сети прямого распространения. Они характерны отсутствием обратных связей в своем строении.

1.2.1 Персептрон и многослойный персептрон.

Простейшей архитектурой нейронных сетей является персептрон, который состоит из одного слоя нейронов. Персептрон относится к моделям линейной классификации. Он не способен решать линейно неразделимые задачи. Позже выяснилось, что если «сложить» несколько персептронов друг с другом, то можно избавиться от данного ограничения. Такую сеть назвали многослойным персептроном [7].

Многослойные персептроны могут быть обучены решать задачи распознавания изображений, однако обычно они используются в роли классификатора в конце свёрточных нейронных сетей. На рисунке 1.6 можно увидеть схематичное представление многослойного персептрона.

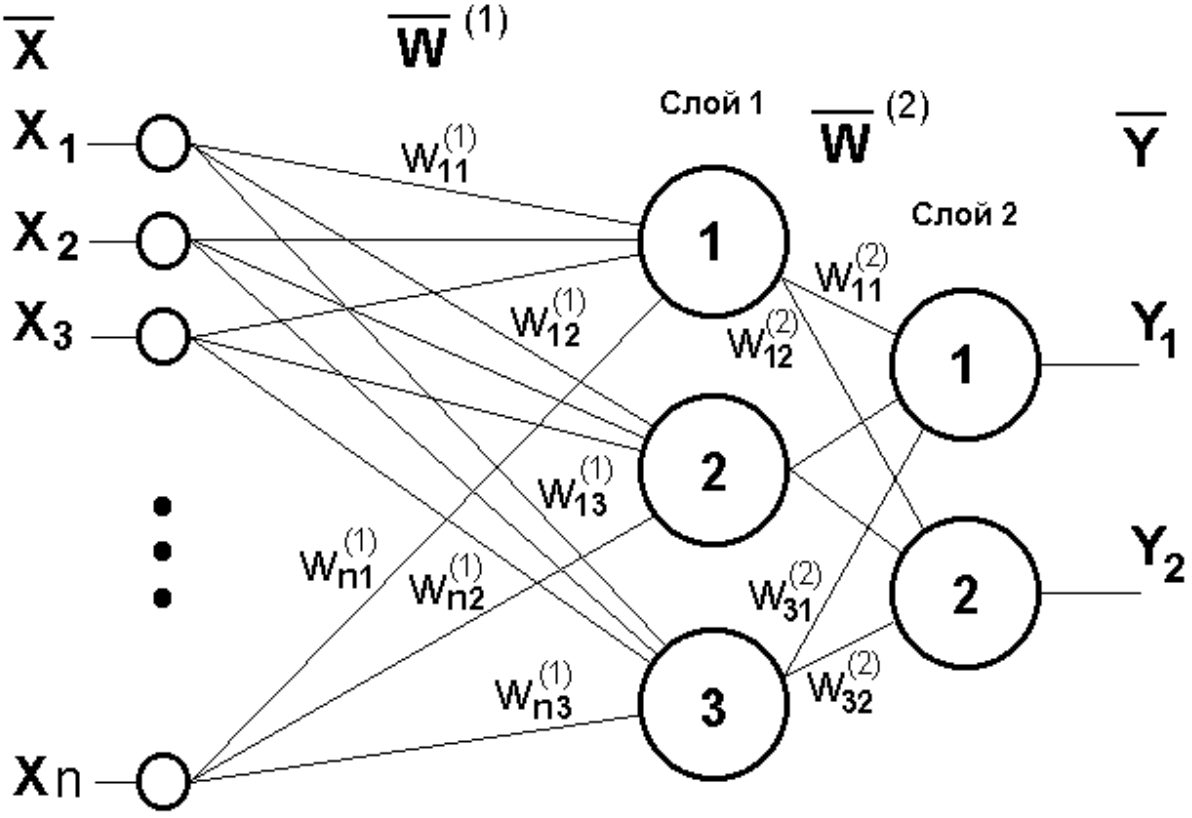

Рис. 1.6. Многослойный персептрон состоящих из двух слоев.

В примере на рисунке 1.6. сеть имеет n-мерный вход, 3 нейрона на первом слое и 2 нейрона на втором. Рассмотрим, как данная сеть будет обрабатывать входные сигналы. Через $w^{(k)}{}_{ij}$ обозначим j-ый вес i-го нейрона k-го слоя. Входные данные можно обозначить через вектор $\overline{X} = (x_1 x_2, \dots, x_n)$. Проходя через нейроны первого слоя, он умножается скалярно на вектор весов нейронов: $(\overline{X}, W^{(1)}{}_i)$, где $W^{(k)}{}_i$ вектор весов i-го нейрона k-го слоя. К результатам умножений применяются функции активации. Полученные числа идут на вход 2-го слоя. Там происходят аналогичные преобразования. В данном случае выход нейронов второго слоя будет являться выходом всей сети, вектором из R^2.

1.2.2 Обратное распространение ошибки.

Сеть должна выдавать корректный выход, поэтому необходимо её обучить. Под обучением сети будем понимать корректировку весов $w^{(k)}{}_{ij}$. Основным способом обучения многослойного персептрона является метод обратного распространения ошибки (англ. backpropagation) [9]. Сначала необходимо вычислить прогноз (выход нейронов выходного слоя). Затем вычисляется ошибка (отклонение текущего выхода от желаемого). Далее начинается подсчет вклада каждого нейрона выходного слоя в общую ошибку. Затем подсчитывается вклад нейронов предпоследнего слоя и т.д. до первого. Т.е. происходит измерение градиента ошибок по всем весам сети, начиная с её конца.

Пронумеруем все нейроны числами от 1 до N. Выход k-го нейрона последнего (выходного) слоя обозначим за o_k. Правильные ответы, которые ожидаем получить на выходах этих нейронов обозначим за y_k. Вес, идущий от i-го нейрона к j-ому, обозначим за w_{ij}. Пусть используется среднеквадратичная ошибка.

$$E = \frac{1}{m}\sum_{k\epsilon \text{Outputs}}(y_k - o_k)^2,$$

где m – число нейронов выходного слоя, Outputs – индексы нейронов выходного слоя.

Также пусть будет сигмоидальная функция активации.

$$f(x) = \frac{1}{1 + e^{-x}}$$

Пошагово алгоритм обратного распространения ошибки будет выглядеть следующим образом:

1. Все веса w_{ij} инициализируются небольшими случайными значениями [2],

Все $\Delta w_{ij} = 0$

2. Следующие шаги повторяем столько раз, сколько обучающих примеров подаем на вход сети:

   2.1. Подаем обучающий пример на вход сети.

   2.2. $\forall k \in$ Outputs: $\delta_k = -o_k(1-o_k)(y_k - o_k)$

   2.3. Далее считаем δ_j для всех нейронов, последовательно

   перемещаясь по слоям от предпоследнего слоя сети к первому.

   $\delta_j = -o_j(1-o_j) \sum_{k \in \text{Children}(j)} \delta_k w_{jk}$,

   где Children(j) – индексы нейронов, на вход которых поступает выход j-го нейрона.

   2.4. Далее для всех весов:

   $$\Delta w_{ij}(n) = -\eta \delta_j o_i$$

   $$w_{ij}(n) = w_{ij}(n-1) + \Delta w_{ij}(n),$$

   где $w_{ij}(n)$ – вес на текущем шаге, $w_{ij}(n-1)$ – вес на предыдущем шаге, η – скорость обучения.

3. Итоговое значение w_{ij} будет новым весом.

1.2.3 Свёрточные нейронные сети.

Свёрточные нейронные сети (англ. convolutional neural networks, CNN) преимущественно используются для распознавания образов и относятся к сетям прямого распространения [5]. В свёрточных нейронных сетях можно выделить два базовых примитива — свёрточный слой (свёртка) и редуцирующий слой (пулинг).

Свёрточный слой состоит из так называемых сверточных ядер. Рассмотрим, что происходит на свёрточном слое на примере чёрно-белого изображения. На вход подается двухмерная матрица значений размерности (n,m). На свёрточном слое также есть рецептивное поле. Рецептивное поле можно

представить как рамку размерности (k,r), которая скользит по матрице с некоторым шагом [6]. Свёрточное ядро представляет из себя матрицу значений той же размерности. На каждом этапе выполняется операция свертки: соответствующие элементы матрицы, попавшие в рецептивное поле, умножаются на соответствующие элементы матрицы сверточного ядра. Далее полученные числа складываются и к полученной сумме применяется функция активации. Итоговое число будет в выходной матрице, которую иногда называют картой признаков. Иллюстрацию данного процесса можно увидеть на рисунке 1.7.

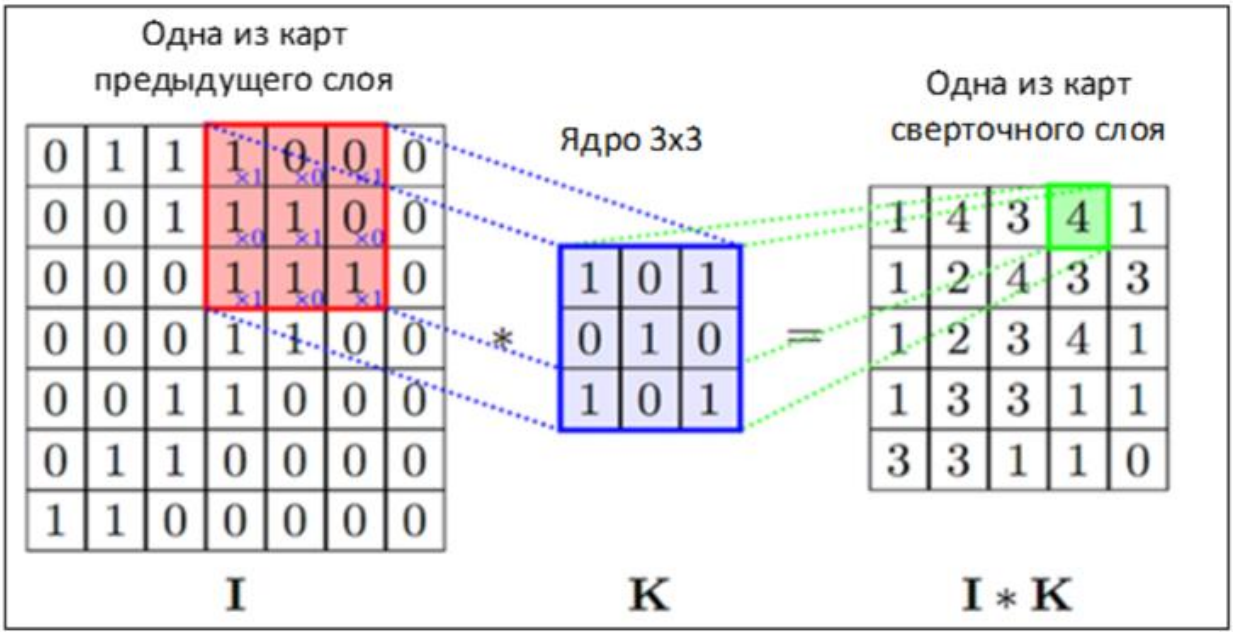

Рисунок 1.7. Операция свёртки

Формула размерности карты признаков, если шаг равен 1:

$$(w, h) = (mW - kW + 1, mH - kH + 1),$$

где w – ширина карты признаков; h – высота карты признаков;

mH – высота исходной матрицы; mW – ширина исходной матрицы;

kH – высота сверточного ядра; kW – ширина сверточного ядра.

Формула свертки:

$$(f * g)[m.n] = \sum_{k,l} f[m - k, n - l] * g[k, l],$$

где f – исходная матрица изображения; g – ядро свертки.

Сколько сверточных ядер будет на сверточном слое, столько же карт признаков будет на выходе этого слоя.

Редуцирующий слой необходим для уменьшения размерности карт признаков предыдущего слоя. Карта признаков, поступающая на вход редуцирующего слоя, фильтруется. Фильтр чаще всего имеет размерность 2x2. Фильтрация происходит следующим образом: карта признаков делится на ячейки 2x2, и выполняется одна из операций. Самые частые – выбор максимального элемента (англ. max pooling) и усреднение (англ. average pooling). В первом случае выходные карты признаков слоя будут состоять из максимальных значений ячеек 2x2, а во втором – из средних арифметических. Визуализацию данного процесса можно увидеть на рисунке 1.8.

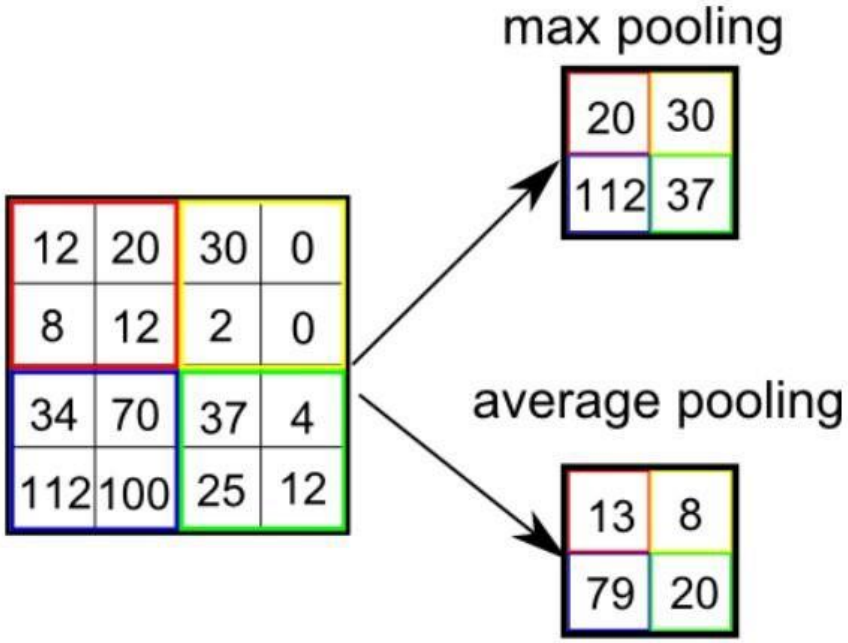

Рисунок 1.8. Пулинг по максимальному и среднему значениям

На последних слоях свёрточных нейронных сетей чаще всего находятся полносвязные слои нейронов, каждый выход элементов предыдущего слоя поступает на входы всем нейронам полносвязного слоя (рисунок 1.9). Соединительным узлом между свёрточными и полносвязными (англ. dense) слоями является слой уплощения (англ. flatten). Он преобразует матрицу (карту признаков) размерности $N \times M$ в вектор из $R^{N \times M}$.

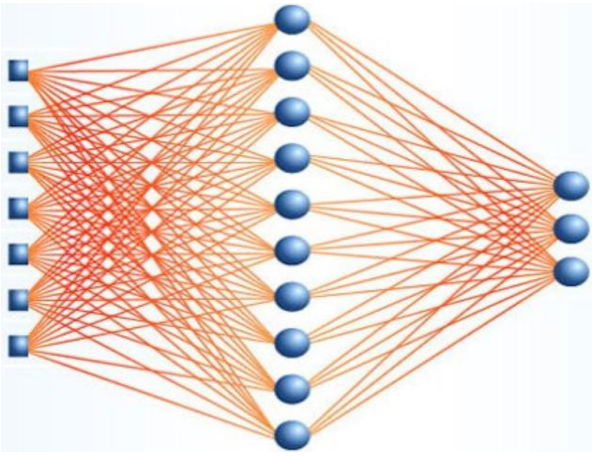

Рисунок 1.9. Полносвязные слои

Обычно искусственные нейронные являются своеобразным «черным ящиком», трудно разложить процесс на этапы так, чтобы было понятно, как формируется выход. Однако преимуществом свёрточных нейронных сетей является возможность визуализации признаков, которые она выделяет на сверточных слоях.

1.3. Сети с обратными связями.

1.3.1 LSTM.

LSTM – сети с долгой краткосрочной памятью. Это один из самых часто используемых видов рекуррентных нейронных сетей, наряду с GRU. В отличие от многих других рекуррентных нейронных сетей, в которых способность связывать информацию теряется с ростом расстояния между элементами, в LSTM информация запоминается на долгое время.

В LSTM ячейке 3 входа и 3 выхода. В момент времени *t* ячейка передает себе же краткосрочное и долгосрочное состояния и принимает те, что были отправлены в момент времени (*t-1).* Третий вход предназначен для обработки данных, поступающих извне. А третий выход предназначен для передачи данных дальше по сети. Рассмотрим рисунок 1.10. и разберемся, как устроена LSTM ячейка [1]. Черным обозначены направления движения данных. Желтым обозначены полносвязные слои нейронов. 3 из них с логистической функцией активации, а оставшийся – с функцией гиперболического тангенса, «x» в красных кружочках – операторы поэлементного умножения, «+» – оператор сложения.

По черной стрелке вверху происходит изменение долгосрочного состояния. Состояние c_t проходит через так называемый шлюз забывания. Там происходит «разрушение» части данных путем поэлементного умножения на выход полносвязного слоя нейронов с логистической функцией активации,

на вход которой подаются краткосрочное состояние h_t и входной сигнал x_t. Затем вектор состояния суммируется с выходом входного шлюза, после чего итоговый вектор состояний c_t идет на выход ячейки. На входном шлюзе происходит аналогичное «разрушение» части сигнала, который получен прохождением состояния h_t и выходного сигнала x_t через полносвязный слой нейронов с гиперболическим тангенсом на выходе. Помимо этого, есть выходной шлюз, который «разрушает» часть преобразованного гиперболическим тангенсом состояния c_t. Получившийся вектор подается на 2 выхода: как краткосрочное состояние в следующий момент времени и как выходной сигнал, идущий дальше по сети. Таким образом, LSTM ячейка способна забывать/запоминать и обновлять свою внутреннюю память.

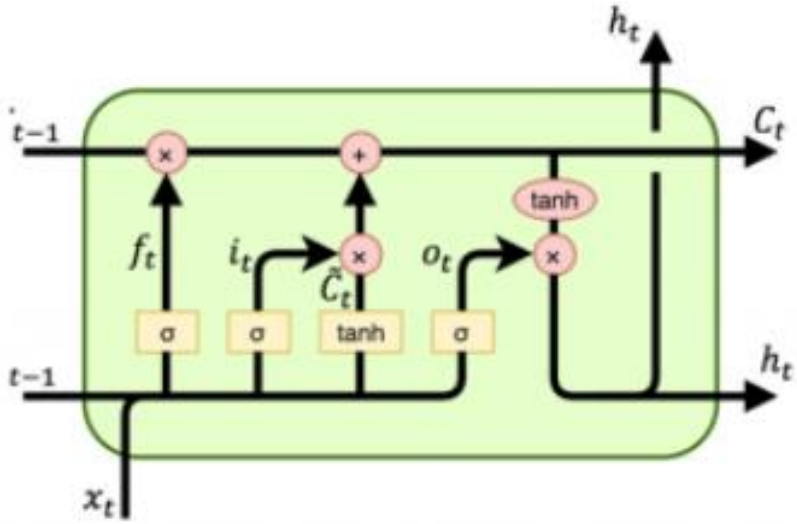

Рисунок 1.10. LSTM ячейка

1.3.2 GRU.

GRU (ячейка управляемого рекуррентного блока) похожа на LSTM по своей сути, но использует меньше операций для вычисления [1]. В отличие от LSTM, в GRU только по 2 входа и 2 выхода. Вместо двух состояний ячейки LSTM в GRU всего один вектор состояний h_t. На рисунке 1.11. представлена схема устройства ячейки GRU.

Принцип работы ячейки примерно тот же: состояние h_{t-1} и входной сигнал x_t поступают на вход полносвязным слоям нейронов с логистической функцией активации, выходные векторы поступают в шлюзы и отвечают за то, какая часть сигналов пройдет дальше. Кроме того, h_{t-1} и x_t поступают на вход слою нейронов с гиперболическим тангенсом. Красный овал с «1-» обозначает вычитания вектора z_t из вектора единиц. Если на один шлюз поступает вектор с компонентой 0, то на второй поступит вектор с компонентой 1 и наоборот. Т.к. выходы этих шлюзов складываются и передаются дальше, это можно интерпретировать как стирание старой информации из состояния h_t перед записью новой.

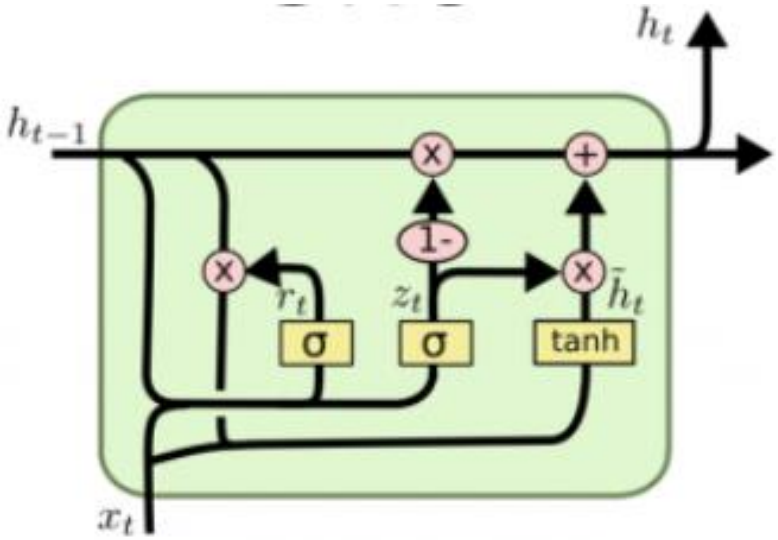

Рисунок 1.11. GRU ячейка

GRU работает быстрее, но из-за упрощенной архитектуры хуже запоминает информацию, поэтому подходит для решения более простых задач.

Глава 2. Постановка задачи. Описание набора данных.

2.1. Набор данных и постановка задачи.

Задача распознавания образов представляет из себя правильное определение класса объекта по его изображению. Изображение представляется в виде одной (для черно-белых), либо трёх (для цветных) матриц значений яркости пикселей. Для цветных изображений каждая матрица отвечает за один из цветов RGB — красный, зеленый и синий. Чем сложнее объекты классификации, тем больше изображений должна «увидеть» сеть, чтобы корректно их классифицировать.

Совокупность таких изображений называют выборкой (датасетом). В данной задаче мы имеем некоторый набор цветных изображений, каждый из которых принадлежит некоторому классу. Данный набор поделен на 2 множества — обучающее и тестовое. Задача состоит в том, чтобы построить некоторую вычислительную модель, которая при помощи изображений обучающего множества научилась бы как можно точнее классифицировать изображения тестового.

Мы используем датасет CIFAR-10. Он состоит из 60000 цветных изображений 32x32 — 50000 тренировочных и 10000 тестовых. На каждой картинке изображен объект одного из следующих классов: самолет, автомобиль, птица, кошка, собака, олень, лягушка, лошадь, корабль, грузовик. Каждый класс представлен 6000 изображениями. Каждому изображению соответствуют 3 матрицы 32x32 с целыми числами от 0 до 255, обозначающими яркость одной из компонент RGB. Каждое изображение помечено одним целым числом от 0 до 9 включительно, каждому числу соответствует свой класс.

На рисунках 2.1., 2.2., 2.3. и 2.4. можно увидеть примеры изображений данного датасета.

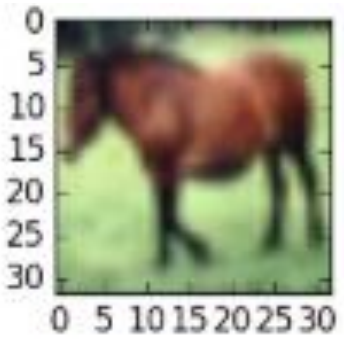

Рисунок 2.1. Лошадь

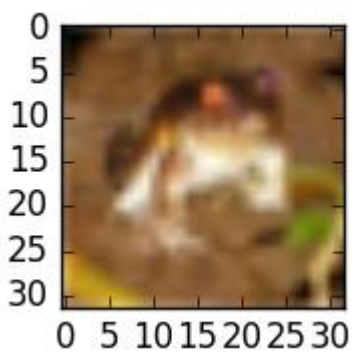

Рисунок 2.2. Лягушка

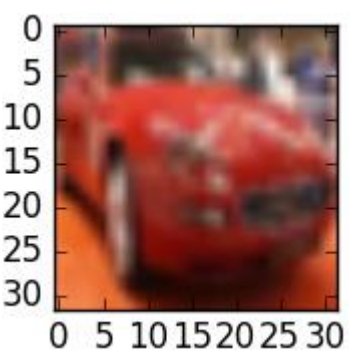

Рисунок 2.3. Автомобиль

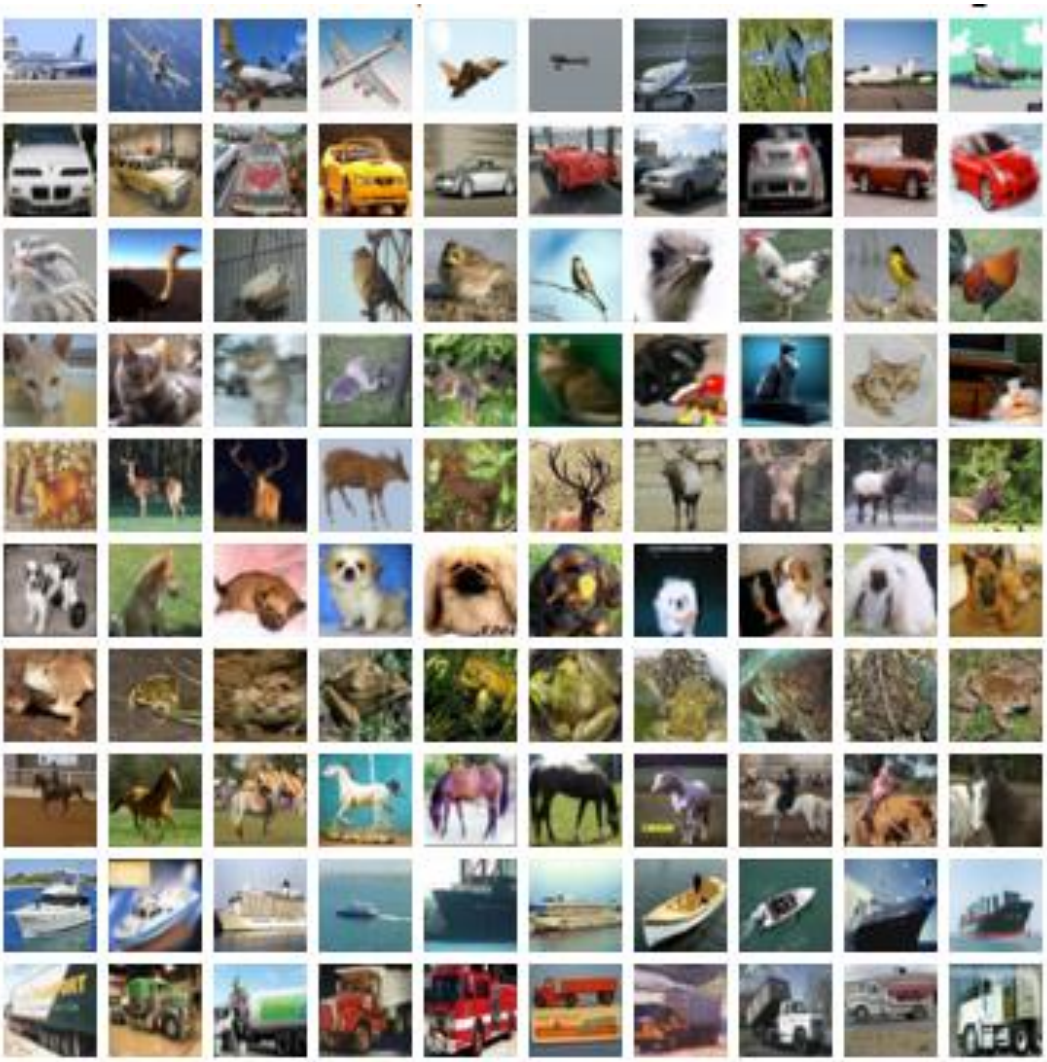

Рисунок 2.4. Примеры изображений CIFAR-10

2.2. Метрики обучения.

Метрики обучения – это чаще всего некоторые числа, которые используют для оценки качества обученной модели. Некоторые примеры метрик:

1. Верность (англ. accuracy).

Является самой простой метрикой, однако не подходит для ситуаций неравномерного распределения элементов выборки по классам.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{P}}{\boldsymbol{N}},$$

где P – количество верных предсказаний; N – число элементов выборки.

Таблица 1. Матрица ошибок.

		Экспертная оценка	
		Положительная	Отрицательная
Оценка сети	Положительная	TP	FP
	Отрицательная	FN	TN

Матрицы ошибок составляются отдельно для каждого класса. В ячейки записывается, сколько раз система дала правильный и неправильный вывод для элементов класса.

2. **Точность (англ. precision).**

$$\boldsymbol{Precision = \frac{TP}{TP + FP}}$$

3. **Полнота (англ. recall).**

$$\boldsymbol{Recall = \frac{TP}{TP + FN}}$$

Точность и полнота используются только для бинарной классификации. Однако, задачу многоклассовой классификации можно свести к задаче бинарной классификации, после чего можно использовать данные метрики.

2.3. Предобработка данных.

Предобработка данных [12] является одним из важнейших этапов решения задач машинного обучения и задач распознавания изображений, в частности. Например, фотографии могут быть зашумлены и иметь разное разрешение. Подавать такие данные на вход сети не имеет смысла.

Изображения CIFAR-10 являются заранее подготовленными, т.е. они не требуют серьезных преобразований. Было проделано лишь несколько манипуляций:

1) Метки от 0 до 9 превратили в единичные вектора из R^{10} – 0 преобразуется в $(1,0,0,\dots,0)$, 1 – в $(0,1,0,\dots,0)$, … , 9 – в $(0,0,0,\dots,1)$. На выходном слое наших нейронных сетей всегда будут 10 нейронов с выходной функцией softmax, т.е. выходом будет являться вектор из R^{10}

вероятностей принадлежности изображения к классам. Например, если на выходе будет вектор (0.1, 0.7, 0, 0.2, 0, 0, 0, 0, 0, 0), это можно будет интерпретировать, что с вероятностью 0.1 оно принадлежит к 1 классу, с вероятностью 0.7 ко второму классу и с вероятностью 0.2 к четвертому классу. Метки должны быть той же размерности, чтобы можно было вычислить функцию ошибки.

2) Значения матриц яркостей пикселей по умолчанию принадлежат целочисленному типу. Они были преобразованы к типу с плавающей точкой, чтобы не возникало проблем с вычислениями.

3) Чтобы улучшить сходимость градиентного спуска, необходимо произвести нормализацию входных данных. Для этого все значения матриц яркостей были поделены на 255. Таким образом, все значения яркостей стали лежать на отрезке от 0 до 1.

2.4. Параметры обучения.

Под гиперпараметрами обучения будем понимать некоторые параметры, которые влияют на обучение, но выбираются до его начала.

1. Размер батча (англ. batch size).
   Датасет практически всегда слишком велик, чтобы целиком подавать его на вход сети, поэтому его делят на пакеты, которые называют батчами [8]. Размер батча определяет то, сколько изображений будет в одном таком батче. Количество батчей, которое подается на вход сети каждую эпоху, называется количеством итераций. Чем меньше итераций, тем меньше времени занимает каждая эпоха. Чем больше итераций, тем больше изменений весов происходит каждую эпоху. Размер батча подбирают эмпирически.

2. Оптимизатор.
   Под оптимизатором понимается метод вычисления градиента. Примеры оптимизаторов: RMSprop, SGD, Adam. Описание этих и многих других оптимизаторов можно найти в [10].

3. Количество эпох обучения.
   Эпоха обучения – полный проход датасета в прямом и обратном направлении. Если эпох будет слишком мало, сеть останется недообучена, если слишком много – переобучена.

4. Скорость обучения (англ. learning rate).
   Параметр, отвечающий за то, насколько сильно будут изменяться веса каждую итерацию. Зависимость можно представить следующей формулой. (Пример можно увидеть в описанном выше алгоритме обратного распространения ошибки)

   $W^n = W^{n-1} - lr * grad,$
   где W^n – новый вес, W^{n-1} – старый вес, lr – скорость обучения, $grad$ – градиент.

2.5. Проблема переобучения сетей.

Одна из часто возникающих проблем при обучении сети – её переобучение [11]. Под переобучением понимают явление, когда сеть чрезмерно «подстраивается» под тренировочную выборку, распознавая её значительно лучше, чем тестовую выборку. Т.е. ухудшается способность сети к обобщению.

Существует несколько методов борьбы с проблемой переобучения, рассмотрим два из них:

1. Ранняя остановка

Стартовые значения весов обычно инициализируются некоторыми малыми значениями, поэтому число «эффективных» весов, т.е. тех, которые вносят ощутимый вклад в выход, мало. С каждой эпохой по мере изменения весов, это число будет возрастать, пока не станет равным числу весов в сети, что приведёт к её переобучению.

Суть метода заключается в том, чтобы вовремя остановить процесс обучения. Задается параметр patience, равный целому числу, например, 5-ти. Когда функция ошибки на валидационном множестве (стр. 21) не будет уменьшаться 5 эпох подряд, обучение останавливается.

2. Dropout

Dropout относится к методам обучения. Он позволяет игнорировать определенную долю p нейронов на слое каждую эпоху обучения. Такие нейроны не будут учитываться при вычислениях во время прямого или обратного прохода. Он позволяет нейронам меньше «полагаться на своих соседей», увеличивая их индивидуальную мощность.

Глава 3. Вычислительный эксперимент. Выводы.

Сети реализованы и обучены при помощи библиотеки Tensorflow [4] и с использованием высокоуровневого языка Python, который на данный момент является самым часто используемым языком для написания искусственных нейронных сетей. Tensorflow – библиотека компании Google, помогающая в реализации и обучении моделей машинного обучения, в том числе нейронных сетей. Обучение происходило на облачном сервисе Google Colab, предоставляющем пользователям бесплатную аренду серверов с видеокартами Tesla K80.

Для сравнения были выбраны 4 архитектуры искусственных нейронных сетей: 2 свёрточные и 2 рекуррентные.

3.1. Выбранные параметры.

Чтобы оценить ту или иную архитектуру искусственной нейронной сети, необходимо перебрать огромное количество гиперпараметров обучения, т.к. они вносят существенный вклад в качество итоговой модели. Существует большое количество методов оптимизации гиперпараметров [3], однако они требуют серьезных вычислительных мощностей, а следовательно, и времязатрат. Поэтому было принято решение перебрать некоторое множество параметров вручную.

Сети обучались с перебором следующих параметров обучения: оптимизатор, кол-во эпох, скорость обучения и параметр исключения слоя dropout. Таблицы с параметрами и полученные верности на тестовом множестве будут приведены отдельно для каждой сети.

Т.к. датасет является сбалансированным по числу количеству изображений каждого класса, будем использовать метрику верности. Для всех сетей устанавливаем размер батча в 128 изображений.

3.2. Валидационное множество.

Тестовое множество нельзя использовать при обучении, однако необходимо каким-то образом контролировать обобщающую способность нейронной сети. Для этой цели от обучающего множество отделяют некоторую часть (обычно 10% или 20%). Эта отделенная часть называется валидационным множеством. Оно не используется для изменения весов, но каждую эпоху оно используется для вычисления верности (или других метрик). В нашем случае под валидационное множество выделяется 20% обучающего множества, т.е. 10000 изображений.

Оптимальное количество эпох находим по лучшей верности валидационном множестве. Именно для этого числа эпох будет вычисляться верность на тестовом множестве.

3.3. Уверенность (Confidence).

Как упоминалось ранее, функция активации softmax превращает вектор выходных значений в вектор вероятностей принадлежности к классу, сумма компонент вектора равна единице. Представим ситуацию: сеть сделала предсказание для изображения, которое принадлежит первому классу. Выходной вектор получился следующего вида: (0.11, 0.09, 0.1, 0.1, 0.1, …, 0.1). Если выбирать наибольшую компоненту вектора (0.11), то получится, что сеть сделала верное предсказание. Но это также значит, что сеть «уверена», в своем выборе лишь на 0.11, т.е. с вероятностью 0.89 изображение может оказаться изображением любого другого класса. Очевидно, что такое предсказание нельзя считать правильным. Поэтому вводят параметр confidence, наибольшая компонента вектора должна превосходить его, чтобы предсказание считалось верным. В нашей задаче параметр confidence принимаем равным 0.5.

3.4. Реализованные архитектуры.

3.4.1 LeNet-5.

LeNet является одной из классических архитектур свёрточных нейронных сетей. Впервые она была предложена еще в 1998 году. Хорошо показала себя на наборе данных MNIST. Схема сети представлена на рисунке 3.1.

Сеть содержит 83 126 обучаемых параметров.

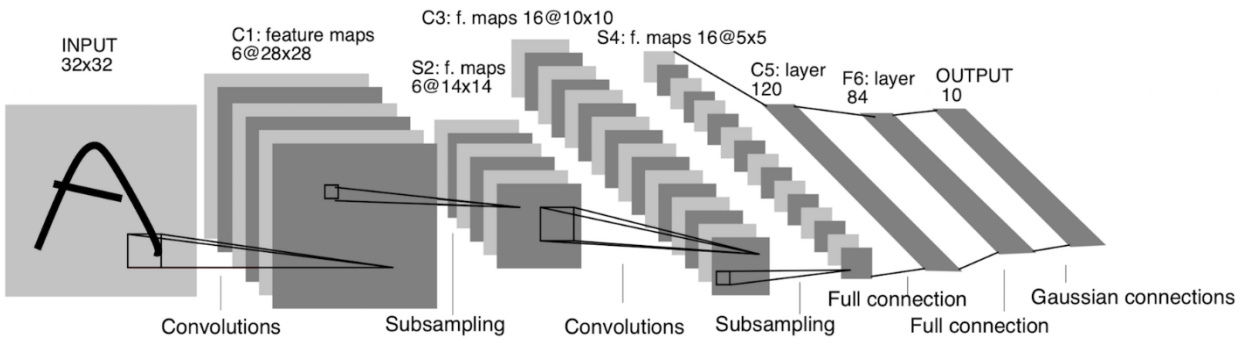

Рисунок 3.1. Свёрточная нейронная сеть LeNet-5.

Таблица 2. Вариация гиперпараметров LeNet-5.

	Оптимизатор	Кол-во эпох	Скорость обучения	Верность
1.	SGD	91	0.01	0.5001
2.	RMSprop	39	0.001	0.565
3.	**RMSprop**	**88**	**0.0005**	**0.5864**
4.	Adam	78	0.0003	0.5648
5.	Adam	195	0.0001	0.5233

Таблица 3. Архитектура сети LeNet-5.

тип слоя	размер ядра	входной размер	выходной размер
Convolutional	5x5	3x32x32	6x32x32
Average pooling	2x2	6x32x32	6x16x16
Convolutional	5x5	6x16x16	16x12x12
Average pooling	2x2	16x12x12	16x6x6
Dense	1x1	1x576	1x120
Dense	1x1	1x120	1x84
Output dense	1x1	1x84	1x10

3.4.2 VGG.

Архитектура свёрточной нейронной сети VGG-16 показала хорошие результаты при распознавании ImageNet, упомянутого в введении. Её основная идея заключается в чередовании слоев группами: два или три свёрточных слоя с рамкой 3x3, а затем слой пулинга со средним значением с рамкой 2x2. Так как размерность изображений CIFAR-10 значительно меньше, архитектуру пришлось скорректировать, убрав 2 группы слоев. Это нужно потому, что уже после третьего слоя пулинга размерность карт признаков становится 4x4, а после четвертого пулинга свёртка с рамкой 3x3 была бы неосуществима. Схема VGG-16 представлена на рисунке 3.2.

Сеть содержит 14 025 866 обучаемых параметров.

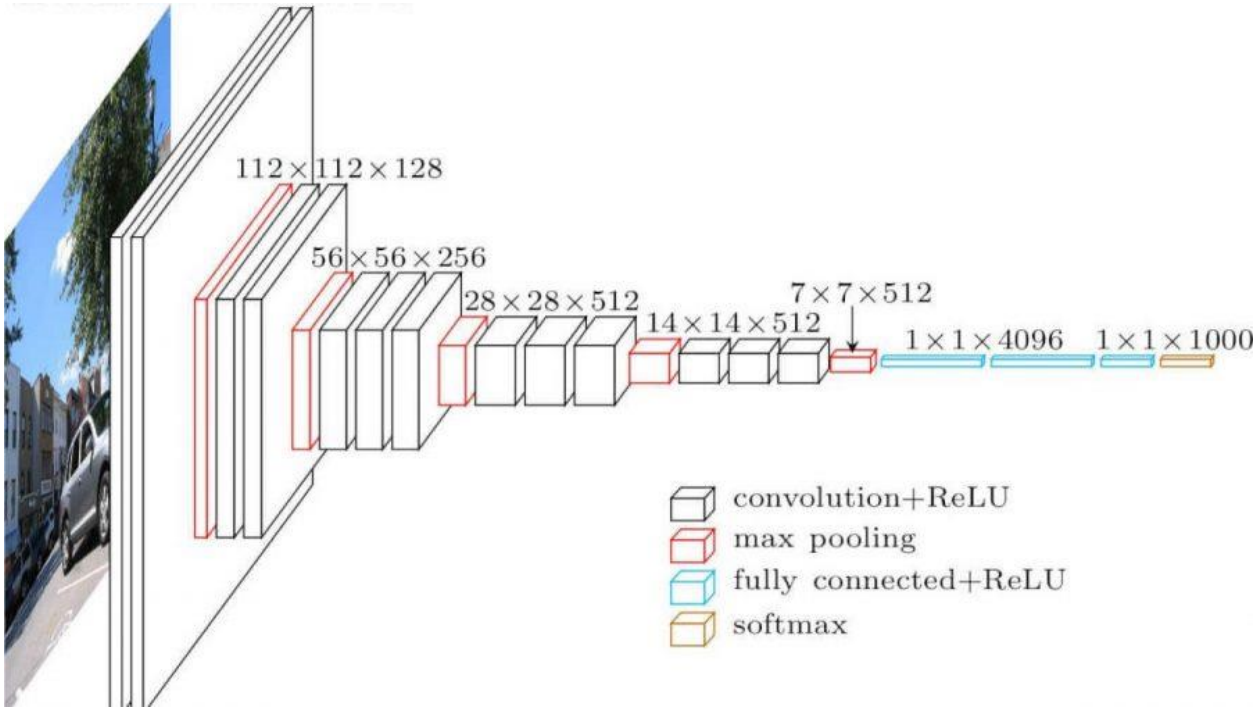

Рисунок 3.2. Свёрточная нейронная сеть VGG-16.

Таблица 4. Вариация гиперпараметров VGG.

	Оптимизатор	Кол-во эпох	Скорость обучения	Dropout	Верность
1.	SGD	68	0.01	0.2	0.6415
2.	RMSprop	35	0.001	0.4	0.7601
3.	RMSprop	37	0.001	0.2	0.7823
4.	Adam	56	0.0005	0.2	0.8037
5.	**Adam**	**63**	**0.0003**	**0.15**	**0.8075**

Таблица 5. Архитектура сети VGG.

тип слоя	размерность ядра	размерность входа	размерность выхода
Convolutional	3x3	3x32x32	128x32x32
Convolutional	3x3	128x32x32	128x32x32
Max-pooling	2x2	128x32x32	128x16x16
Convolutional	3x3	128x16x16	256x16x16
Convolutional	3x3	256x16x16	256x16x16
Max-pooling	2x2	256x16x16	256x8x8
Convolutional	3x3	256x8x8	512x8x8
Convolutional	3x3	512x8x8	512x8x8

Max-pooling	2x2	512x8x8	512x4x4
Dense	1x1	512x4x4	1x1024
Dropout	-	-	-
Dense	1x1	1x1024	1x1024
Dropout	-	-	-
Output dense	1x1	1x1024	1x10

3.4.3 Сеть с LSTM ячейками.

Данная архитектура была выбрана потому, что показала хорошие результаты на наборе данных MNIST.

Сеть содержит 646 154 обучаемых параметров.

Таблица 6. Вариация гиперпараметров LSTM.

	Оптимизатор	Кол-во эпох	Скорость обучения	Dropout	Верность
1	RMSprop	66	0.001	0.2	0.5816
2	Adam	112	0.0005	0.3	0.5836
3	RMSprop	140	0.0005	0.1	0.5757
4	**Adam**	**67**	**0.001**	**0.1**	**0.5982**
5	SGD	228	0.01	0.2	0.5262

Таблица 7. Архитектура сети LSTM.

тип слоя	размерность входа	размерность выхода
LSTM	32x96	32x128
Dropout	-	-
LSTM	32x128	1x256
Dropout	-	-
Dense	1x256	1x512
Dropout	-	-
Output dense	1x512	1x10

3.4.4 Сеть с GRU ячейками.

Как упоминалось ранее, GRU ячейки похожи на LSTM. Кроме того, они являются взаимозаменяемыми. Для более удобного сравнения реализуемая архитектура была выбрана точно такой же с точностью до замены ячеек LSTM на ячейки GRU.

Сеть содержит 519 946 обучаемых параметров

Таблица 8. Вариация гиперпараметров GRU.

	Оптимизатор	Кол-во эпох	Скорость обучения	Dropout	Верность
1	RMSprop	66	0.001	0.2	0.6186
2	SGD	91	0.1	0.3	0.5822
3	Adam	130	0.0005	0.2	0.6225
4	**Adam**	**221**	**0.0003**	**0.15**	**0.6295**
5	RMSprop	105	0.0005	0.1	0.6195

Таблица 9. Архитектура сети GRU.

тип слоя	размерность входа	размерность выхода
GRU	32x96	32x128
Dropout	-	-
GRU	32x128	1x256
Dropout	-	-
Dense	1x256	1x512
Dropout	-	-
Output dense	1x512	1x10

Таблица 10. Верность предсказаний.

Множество, на котором измеряется верность / Название Архитектуры	Тестовое множество	Тренировочное множество	Валидационное множество
LeNet-5	0.5655	0.89425	0.5884
VGG	0.8075	0.998575	0.817
LSTM	0.5982	0.99175	0.6048
GRU	0.6295	0.992275	0.6307

3.5 Вычислительный эксперимент

Для визуализации того, как количество эпох влияет на результативность модели, были построены графики зависимости верностей на тренировочном, тестовом и валидационном множествах от числа эпох. На рисунке 3.1. можно увидеть график для сети LeNet-5, на рисунке 3.2. — для VGG, на рисунке 3.3. для LSTM и на рисунке 3.4. — для GRU.

Рисунок 3.3. График зависимости верности предсказаний от числа эпох для сети LeNet-5

Рисунок 3.4. График зависимости верности предсказаний от числа эпох для сети VGG

Рисунок 3.5. График зависимости верности предсказаний от числа эпох для сети LSTM

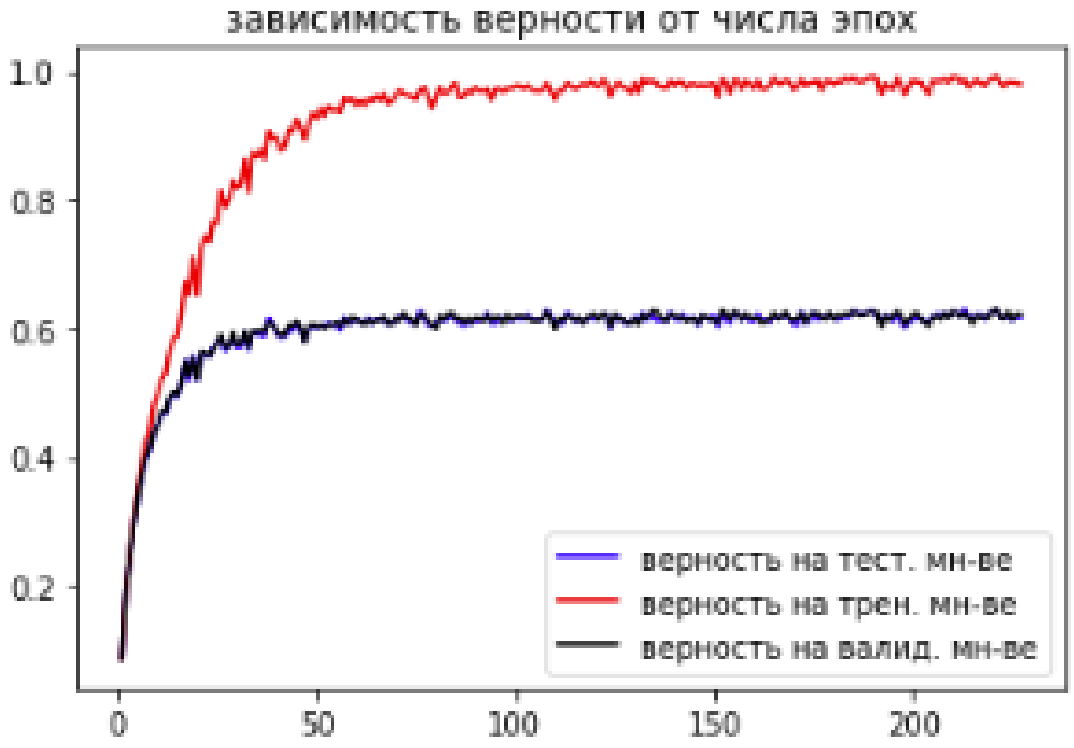

Рисунок 3.6. График зависимости верности предсказаний от числа эпох для сети GRU

На графиках видно, что VGG гораздо быстрее сходится к своей лучшей точности, которую можно достичь на выбранных параметрах обучения. Однако, VGG имеет в 168 раз больше параметров, что даже при учете более быстрой сходимости модели требует значительно большего времени на обучение.

3.6 Выводы

На основании проделанной работы можно сформулировать основные выводы.

Была проведена предобработка изображений CIFAR-10. Реализованы 4 искусственных нейронных сети: сверточные сети Lenet-5 и VGG и рекуррентные сети LSTM и GRU. Сети были обучены на тренировочном множестве из 40000 изображений с валидационным множеством из 10000 изображений. Для каждой сети перебирались гиперпараметры обучения, среди которых были выбраны лучшие. Лучшая верность предсказания на тестовом множестве была получена сетью VGG, она составила 0.8075 с параметром уверенности равным 0.5. Она была получена при следующих параметрах:

– оптимизатор Adam

– 63 эпохи обучения

– learning rate (скорость обучения) = 0.0003

– вероятность исключения dropout = 0.15

Худшая же верность была получена сетью LeNet-5, она составила 0.5655. Это можно объяснить тем, что свёрточная сеть LeNet-5 имеет слишком малое число обучаемых параметров для решения поставленной задачи: VGG содержит 14 025 866 обучаемых параметров, а LeNet-5 лишь 83 126. Однако аналогичные по структуре сети с LSTM и GRU ячейками показали, что не всегда большее число параметров приводит к лучшим результатам. GRU, являясь более простой сетью, чем LSTM, показала лучшую верность предсказаний: 0.6295 против 0.5982. При этом сеть LSTM содержит 646 154 обучаемых параметров, а сеть GRU – 519 946. Интересно то, что рекуррентные нейронные сети, предназначенные для задач другого типа, справились с распознаванием изображений на уровне со свёрточными нейронными сетями.

В 3 сетях из 4 наилучшую верность обеспечил оптимизатор Adam. Оптимальной вероятностью исключения dropout оказались значения от 0.1 до 0.15. При бóльших значениях данного параметра сети недообучались.

Заключение.

Таким образом, при выполнении данной работы были выполнены следующие задачи:

1. Предобработка набора изображений CIFAR-10.
2. Реализация четырех архитектур искуственных нейронных сетей: LeNet-5, VGG, LSTM и GRU.
3. Обучение сетей распознаванию набора изображений CIFAR-10 с перебором различных параметров обучения.
4. Анализ и сравнение полученных результатов.

Список литературы.

1. Орельен Жерон. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем / / пер. с англ. – СПб.: ООО «Альфа-книга», 2018. – 688 с.

2. Умберто Микелуччи. Прикладное глубокое обучение. Подход к понимаю глубоких нейронных сетей на основе метода кейсов. / / пер. с англ. – СПб.: БХВ-Петербург, 2020. – 368 с.

3. Бхарат Рамсундар, Реза Босаг Заде. TensorFlow для глубокого обучения. От линейной регрессии до обучения с максимизацией подкрепления. / / пер. с англ. – СПб.: БХВ-Петербург, 2019. – 256 с.

4. Антонио Джулли, Суджит Пал. Библиотека Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow / / пер. с англ. – М.: ДМК Пресс, 2018. – 294 с.

5. Николенко С., Кадурин Е., Архангельская А. Глубокое обучение. Погружение в мир нейронных сетей. – СПб, 2018. – 480 с.

6. Джулли А, Пал С. Библиотека Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow / / пер. с англ. М.: ДМК Пресс, 2018. – 294 с.

7. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / / пер. с англ. М.: ДМК Пресс, 2015. – 400 с.

8. Шолле Ф. Глубокое обучение на Python / / пер. с англ. СПб.: Питер, 2018. – 400 с.

9. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение. – Москва: ДМК-Пресс, 2017. – 652 с.

10. An overview of gradient descent optimization algorithms, https://ruder.io/optimizing-gradient-descent/index.html.

11. Траск Э. Грокаем глубокое обучение / / пер. с англ. СПб.: Питер, 2019. – 352 с.

12. Джоши, Пратик. Искусственный интеллект с примерами на Python / / пер. с англ. – СПб.: ООО «Диалектика», 2019. – 448 с.