МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра Электроника и робототехника

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой к.т.н., доцент Чигамбаев Т.О.

_____«____» _____ 2020 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

На тему: «Применение искусственных нейронных сетей в противовоздушной обороне»

Специальность 6М071600 – «Приборостроение»

Выполнил: Рахметуллаев М.А. Группа МПсн-18-1

Научный руководитель: PhD, доцент Балбаев Г.К.

_____«____» _____ 2020 г.

Нормоконтролер: к.т.н., доцент Чигамбаев Т.О.

_____«____» _____ 2020 г.

Рецензент: к.т.н., ассоциированный профессор (доцент) Международный Университет Информационных Технологий Иманбекова Т.Д.

_____«____» _____ 2020 г.

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество

«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт космической инженерии и телекоммуникаций

Кафедра Электроника и робототехника

Специальность 6М071600 – «Приборостроение»

ЗАДАНИЕ

на выполнение магистерской диссертации

Магистранту Рахметуллаев Мухаммед Абумуталлапулы

Тема проекта: Применение искусственных нейронных сетей в противовоздушной обороне

Утверждена приказом по университету № 122 от «25» ___10____ 2018 г.

Срок сдачи законченного проекта «10» ___06___ 2020 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Техническая документация по языку Python, Python, библиотека TensorFlow, операционная система Windows 10.

Перечень вопросов, подлежащих разработке в диссертационной работе, или краткое содержание диссертационной работы:

а) обзорно-аналитическая часть;

б) обзор языков программирования и фреймворков;

в) моделирование нейроуправления;

г) машинное обучения в классификации радиолокационных целей.

Основная рекомендуемая литература:

1 А.В.Нестеров «Анализ методов цифровой обработки информации в системах компьютерного зрения и обзор областей применения данных систем», ISSN 1995-4565. Вестник РГРТУ. № 4 (выпуск 26). Рязань, 2008

2 Dr. Romik Chatterjee. `Advanced Color Machine Vision and Applications`, Graftek Imaging Inc. 2014

3 Gary Bradski, Adrian Kaehler/ Learning OpenCV: Computer Vision with the OpenCV Library – Соросовский образовательный журнал, с. 118-124, 1996 - ISBN:978-0-596-51613-0 July 22, 2011

4 Saurabh Kapur/ Computer Vision with Python 3 – Packt Publishing Ltd. Birmingham, August 2017 - ISBN 978-1-78829-976-3

Консультации по проекту с указанием относящихся к ним разделов работы

Раздел	Консультант	Сроки	Подпись
Теоретическая часть	Балбаев Г.К.	19.10.2018-25.03.2019	
Программная часть	Балбаев Г.К.	15.04.2019-16.09.2019	
Конструкторская часть	Балбаев Г.К.	07.11.2019-25.05.2019	

График
подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Обзорно-аналитическая часть	25.03.2019	
Обзор языков программирования и фреймворков	16.09.2019	
Моделирование нейроуправления	2.12.2019	
Машинное обучения в классификации радиолокационных целей.	25.05.2020	

Дата выдачи задания «15» _____11_____ 2018 г.

Заведующий кафедрой _____ Чигамбаев Т.О.

Научный руководитель проекта _____ Балбаев Г.К.

Задание принял к исполнению магистрант _____ Рахметуллаев М.А.

Аннотация

В данной работе рассматривается применение искусственных нейронных сетей в военной сфере, а именно Кибербезопасность, Радиолокация. Будет рассмотрены методы в Машинного обучения для военных задач, а также будет разрабатываться программа для классификации радиолокационных целей с использованием машинного обучения и глубокого обучения.

Аңдатпа

Бұл жұмыста әскери салада жасанды нейрондық желілерді, атап айтқанда киберқауіпсіздік, радар қолдануды қарастырады. Әскери тапсырмаларды машиналық оқытудың әдістері қарастырылатын болады, сонымен қатар машиналық оқыту мен тереңдетіп оқытуды қолдана отырып, радиолокациялық нысандарды жіктеу бағдарламасы жасалады.

Annotation

This work examines the use of artificial neural networks in the military sphere, namely Cyber Security, Radar. Methods in Machine Learning for military tasks will be considered, and a program will be developed for classifying radar targets using machine learning and deep learning.

Содержание

Введение . . . . . .

1 Введение в Машинное Обучение. . . . . .

1.1 Базовые понятия ИИ . . . . . .

1.2 Машинное обучение . . . . . .

1.3 Нейронные сети в системах управления ……

2 Языки программирования и фреймворки для алгоритмов машинного обучения . . . . . .

2.1 Python, TensorFlow, PyTorch ……. . . . . .

2.2 C++, OpenNN ……. . . . . .

2.3 Matlab ……

3 Классификация радиолокационных целей с использованием машинного обучения и глубокого обучения. . . . . .

3.1 Генерирование синтетических паттернов ……. . . .

3.2 Вейвлет рассеивание . . . . . .

3.3 Обучения модели . . . . . .

3.4 Классификация …….. . . . . .

3.5 Использование сверточных нейронных сетей . . . . . .

3.6 Вейвлет преобразование ……..

3.7 Подготовка изображений

3.8 LSTM

Заключение . . . . . .

Список литературы . . . . . .

Введение

Машинное обучение и глубокое обучение (Machine Learning, Deep Learning) с каждым днем охватывает всё большее место в нашей повседневной жизни ввиду огромного спектра его применений и возможностей. Начиная от анализа и распознавания изображений и заканчивая самоуправляемыми автомобилями, все эти задачи теперь решаются самообучаемыми машинами и алгоритмами. Машинное обучение относят к одной из перспективной области искусственного интеллекта, основная задача которого заключается в том, чтобы компьютерная программа не просто использовала заранее написанный алгоритм, а чтобы система сама обучилась решению поставленной задачи. Согласно результатам международного исследования компании Microsoft, 94% руководителей и топ-менеджеров считают, что технологии ИИ необходимы для решения стратегических задач их организаций. При этом 27% опрошенных уже интегрировали технологии искусственного интеллекта в ключевые бизнес-процессы, еще 45% ведут тестовые проекты. На сегодня искусственные нейронные сети являются мощным инструментом машинного обучения для изучения и решения задач искусственного интеллекта.

Развитие искусственного интеллекта в военной сфере является очень важным направлением во внешней политике таких стран, как США, Китай, Россия, Израиль. Сегодня ИИ уже применяется в симуляторах для военных операций, для распознавания объектов в зоне боевых действий, в распознаваний радиосигналов и в других радиотехнических и радиолокационных задачах.

В конце января 2020 года компания CB Insights провела масштабный анализ тенденций вложения в искусственный интеллект и заявила, что в 2019 году специализирующиеся на технологиях Machine Learning, Deep Learning стартапы привлекли рекордные вложения — $26,6 млрд, заключив более 2200 сделок по всему миру. Для сравнения, в 2018 году было заключено около 1900 соглашений на общую сумму $22,1 млрд, а в 2017-м — около 1700 штук на $16,8 млрд. Зафиксированный аналитиками рекорд соответствует анализу других организаций, следящих за инвестициями в экосистему ИИ. Ранее Национальная ассоциация венчурного капитала сообщила, что хотя общий венчурный капитал в 2019 году снизился, инвесторы по всему миру внесли рекордные $18,4 млрд в развитие и масштабирование ИИ стартапов в США.

1 Введение в машинное обучение

Основная идея проектирования интеллектуальных вычислительных устройств по образу и подобию биологических систем привела к открытию теории искусственных нейронных сетей, ставшей одним из самых мощных и эффективных подходов к разработке искусственного интеллекта. Если говорить о сегодняшних достижениях, то, например, нейронные сети являются основным направлением деятельности компании Deepmind (ныне собственность компании Google), добившейся таких фантастических успехов, как создание нейронной сети, способной учиться играть в видеоигры, и еще одной, которая смогла победить в невероятно сложной игре Go гроссмейстера мирового уровня.

1.1 Базовые понятия ИИ

Термин и понятие «Искусственный Интеллект» появился в 1956 году и ввел его Джон Маккарти, но настоящей популярности технология ИИ достигла лишь сегодня на фоне увеличения объемов данных, усовершенствования алгоритмов и роста вычислительных мощностей. Точного и единого определения ИИ нету, но более подходящее определение гласит так: “Искусственным интеллектом (ИИ) — способность интеллектуальных машин выполнять творческие функции, которые традиционно считаются прерогативой человека.” Также термином ИИ обозначают науку и технологию создания интеллектуальных машин.[1]

Основные свойства ИИ — это понимание языка, обучение и способность мыслить и, что немаловажно, действовать. ИИ – комплекс родственных технологий и процессов, развивающихся качественно и стремительно, например:

1. обработка текста на естественном языке
2. машинное обучение
3. экспертные системы
4. виртуальные агенты (чат-боты и виртуальные помощники)
5. системы рекомендаций.

1.2 Машинное обучение

Машинное обучение (англ. machine learning, ML) — область искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в

цифровой форме. [2]

Различают два типа обучения:

1. Обучение по прецедентам, или индуктивное обучение, основано на выявлении эмпирических закономерностей в данных.
2. Дедуктивное обучение предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний.

Машинное обучение находится на стыке математической статистики, методов оптимизации и классических математических дисциплин, но имеет также и собственную особенность, связанную с проблемами вычислительной эффективности и переобучения. Многие методы индуктивного обучения разрабатывались как альтернатива классическим статистическим подходам. Многие методы тесно связаны с извлечением информации и интеллектуальным анализом данных (Data Mining).

Машинное обучение — не только математическая, но и практическая, инженерная дисциплина. Чистая теория, как обычно, не приводит сразу к решениям, применимыми на практике. Ни одно исследование и тестирование в машинном обучении не обходится без эксперимента на реальных данных, подтверждающего практическую работоспособность метода и эффективную прогнозируемость на новых данных которые не использовались в эксперименте.

Направление машинного обучения, с одной стороны, образовался в результате разделения науки об искусственных нейронных сетях на методы обучения сетей и их архитектуры, с другой стороны — собрал в себе методы обычной математической статистики. Указанные ниже способы машинного обучения исходят из случая использования нейросетей, хотя существуют и другие методы, использующие понятие обучающей выборки — например, дискриминантный анализ, оперирующий обобщённой дисперсией и ковариацией наблюдаемой статистики, или байесовские классификаторы. Базовые типы нейронных сетей, такие как перцептрон и многослойный перцептрон (а также их измененные виды), могут обучаться как с учителем, также и без учителя, с подкреплением и самоорганизацией. Но некоторые виды нейросетей и большинство статистических методов можно отнести только к одному из методов обучения. Поэтому, если необходимо классифицировать методы машинного обучения в зависимости от способа обучения, будет некорректным относить нейросети к определенному виду, правильнее было бы типизировать алгоритмы обучения нейронных сетей. Технически обучение заключается в нахождении коэффициентов (весов) связей между нейронами. В процессе обучения искусственная нейронная сеть может выявлять сложные и комплексные зависимости между входными данными (Input) и выходными (Output), а также выполнять обобщение и регуляризацию. Это означает, что в случае успешного обучения, нейросеть сможет дать верный вывод (output) на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или искаженных данных. С точки зрения математики, искусственные нейронные сети – это

громадный распределенный параллельный процессор, содержащий в себе элементарны единицы обработки информации и это все связано между собой функциональным отношением между входными данными и матрицей весов (коэффициентов). [3]

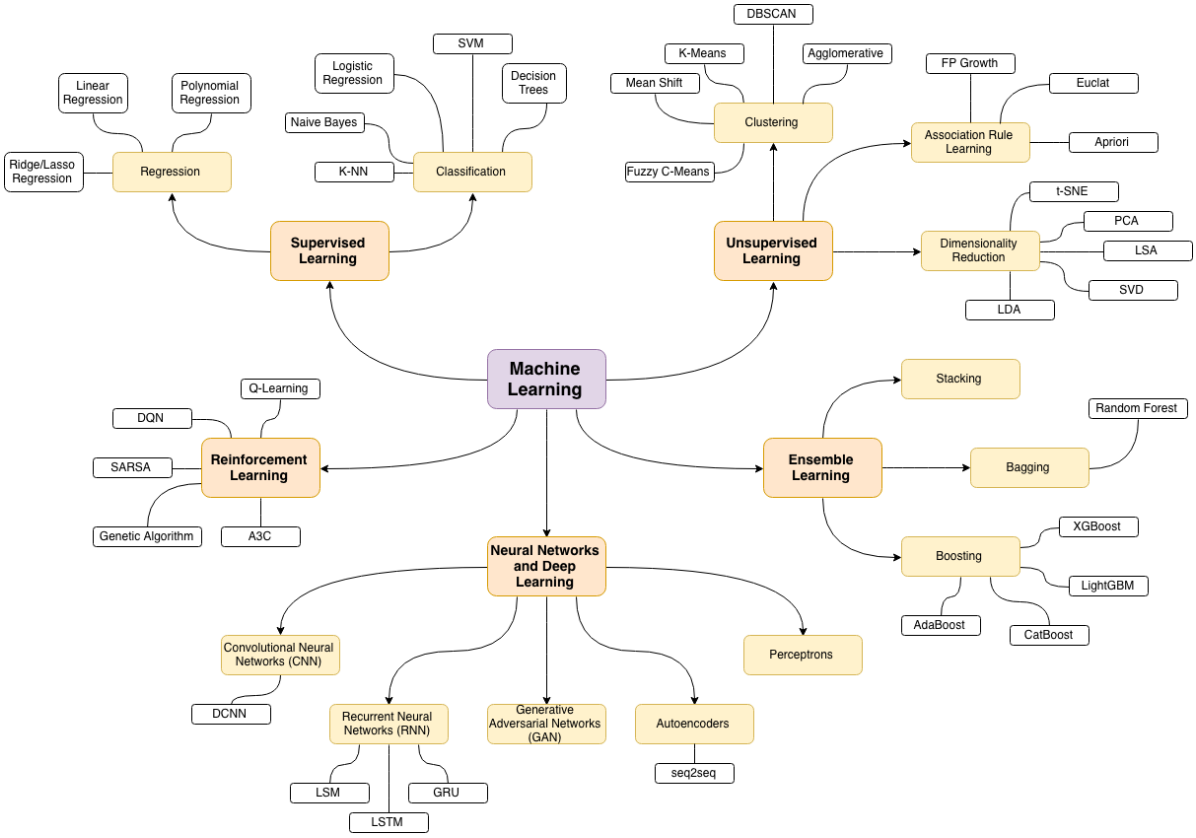

Рисунок 1. Карта машинного обучения

Обучение с учителем (англ. Supervised learning) — один из способов машинного обучения, в ходе которого испытуемая система принудительно обучается с помощью примеров «стимул-реакция». С точки зрения кибернетики, является одним из видов кибернетического эксперимента. Между входами и эталонными выходами (стимул-реакция) может существовать некоторая зависимость, но она неизвестна. Известна только конечная совокупность прецедентов — пар «стимул-реакция», называемая обучающей выборкой. На основе этих данных требуется восстановить зависимость (построить модель отношений стимул-реакция, пригодных для прогнозирования), то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ (рисунок 2). Для измерения точности ответов, так же как и в обучении на примерах, может вводиться функционал качества. Обучение нейронной сети с учителем предполагает, что для каждого входного вектора из обучающего множества данных существует фактическое значение выходного вектора, называемого целевым. Эти вектора образуют обучающую выборку. Веса сети изменяются до тех пор, пока для каждого входного вектора не будет получен необходимый уровень отклонения

выходного вектора от целевого.[4]

Обучение нейронной сети без учителя (Unsupervised learning) является намного более правдоподобной моделью обучения с точки зрения биологических корней искусственных нейронных сетей. Обучающее множество состоит лишь из входных векторов. Алгоритм обучения нейронной сети подстраивает веса сети так, чтобы получались согласованные выходные векторы, т.е. это регрессионная задача. Биологическая нейронная сеть имеет сложные связи, процессы и не имеет обучающей выборки. Современные искусственные нейросети являются упрощенными моделями биологических нейросетей, так как в живом организме присутствуют биохимические и биоэлектрические процессы, которые непросто описать языком программирования.

Типы задач обучения без учителя:

1. Кластеризация
2. Поиск ассоциативных правил
3. Сокращение размерности
4. Визуализация данных

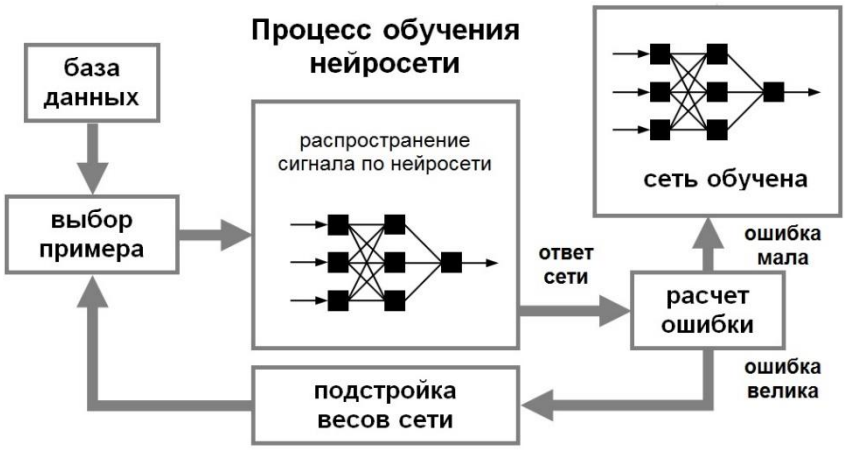

Рисунок 2. Иллюстрация процесса обучения НС

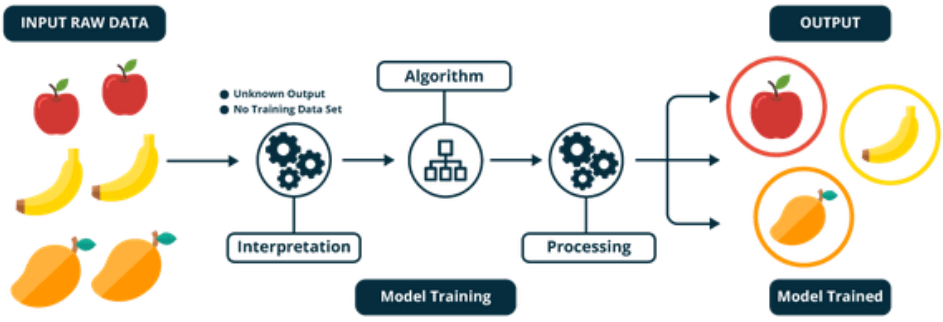

Рисунок 3. Иллюстрация процесса обучения НС без учителя – кластеризация изображений

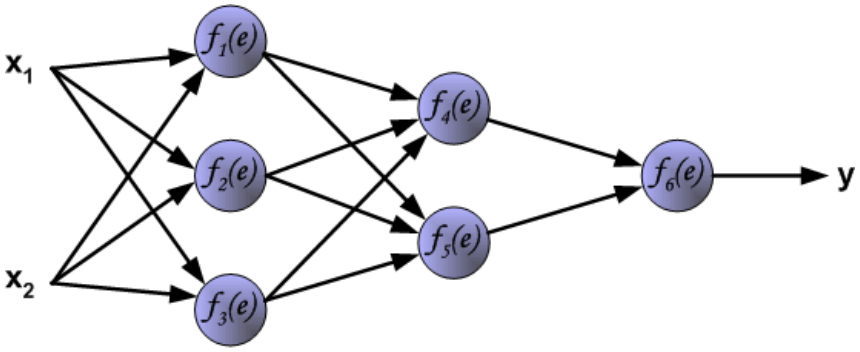

Рисунок 4. Простая нейронная сеть

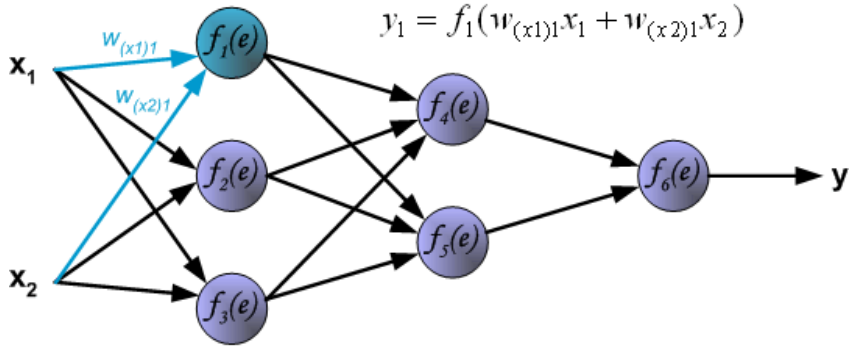

Рисунок 5. Процесс прямого распространения

Обратное распространение ошибки — это способ обучения искусственной нейронной сети. Цели обратного распространения просты: отрегулировать каждый вес пропорционально тому, насколько он способствует общей ошибке. Если итеративно уменьшать ошибку каждого веса, в конце концов у нас будет ряд сбалансированных весов, которые дают хорошие сигналы (прогнозы). [5]

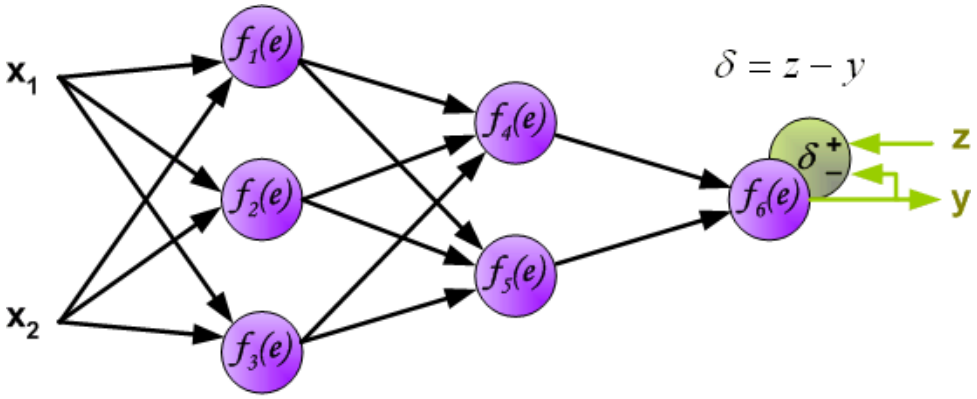

Рисунок 6. Расчет ошибки сети

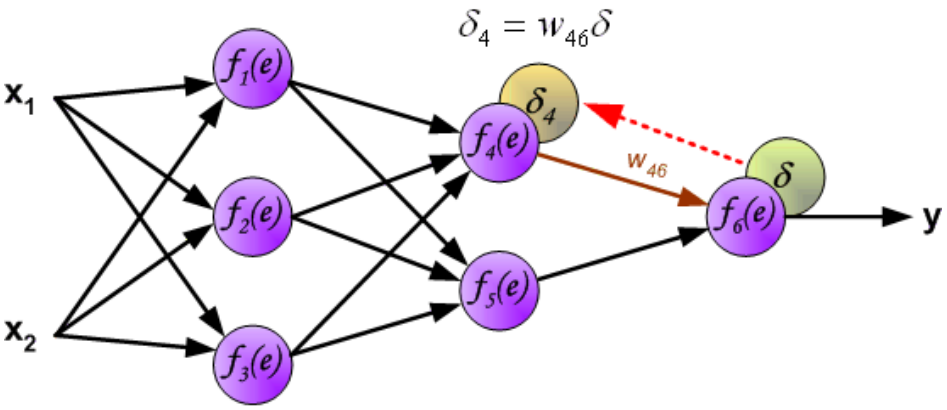

Рисунок 7. Обратное распространение сигнала

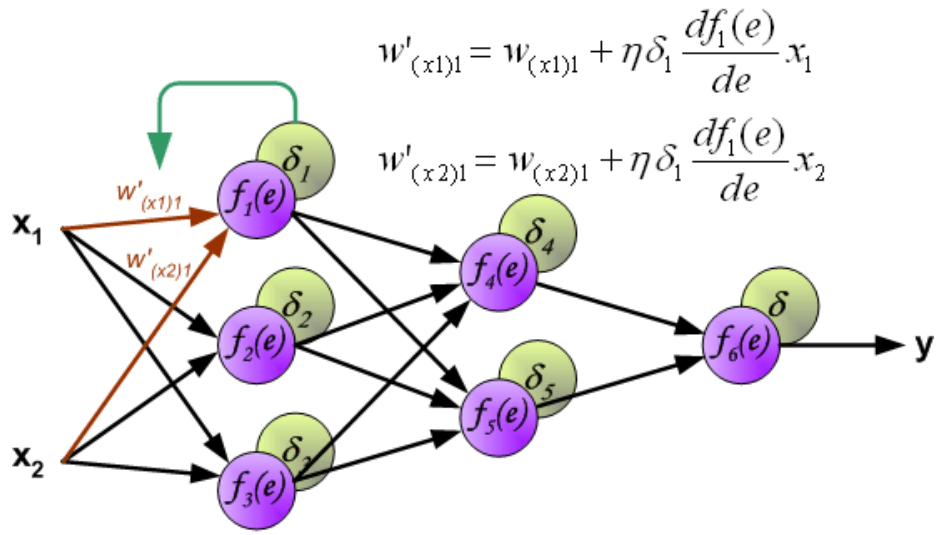

Рисунок 8. Обратное распространение сигнала

Вычисление производной (рисунок 8) необходимо, для корректировки весовых коэффициентов при обучении нейросети с помощью алгоритма обратного распространения сигнала (ошибки), используется метод градиентного спуска. [6]

Градиентный спуск — метод нахождения минимального значения функции потерь (существует множество видов этой функции). Минимизация любой функции означает поиск самой глубокой впадины в этой функции. Функция потерь используется, чтобы контролировать ошибку в валидации модели машинного обучения. Поиск минимальных значений означает получение наименьшей возможной ошибки или повышение точности модели (Accuracy). Точность увеличивается, перебирая набор учебных данных при настройке параметров модели нейросети (весов и смещений).

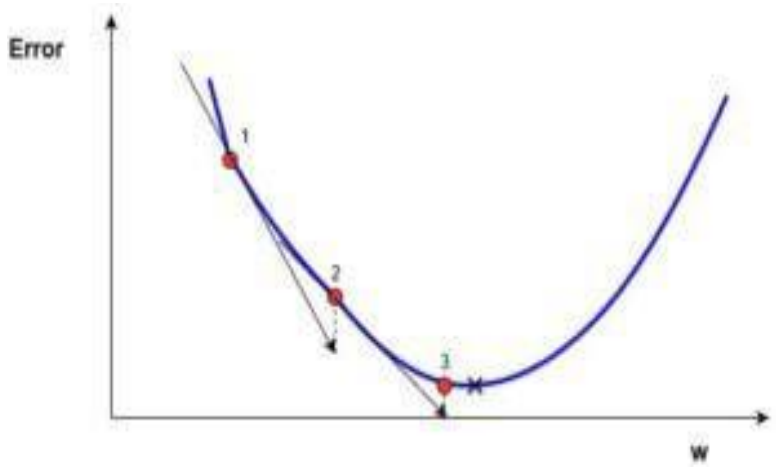

Рисунок 9. Функция потерь

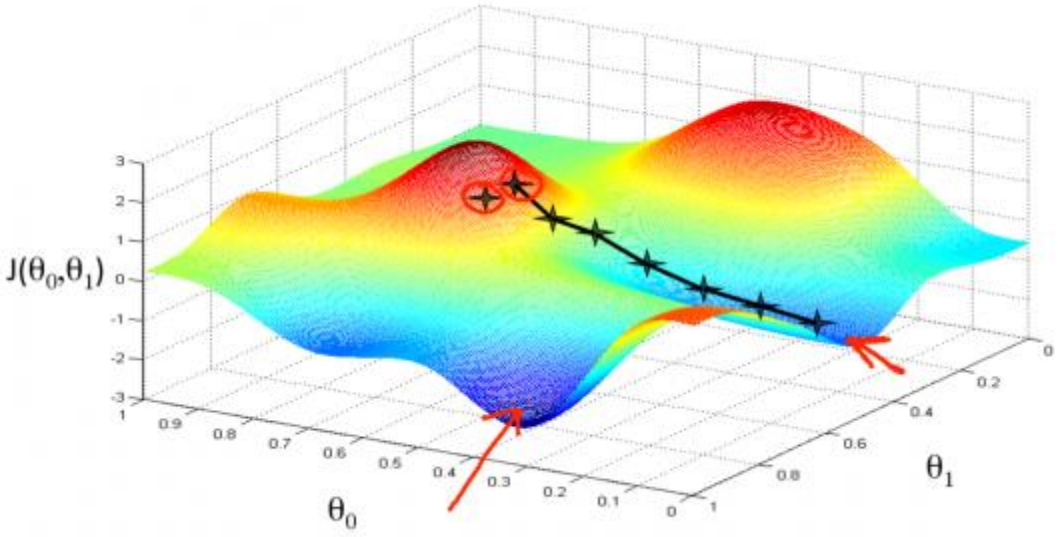

Рисунок 10. Поиск минимума функции

Помимо функции потерь градиентный спуск также требует градиент, который является dJ/dw (производная функции потерь относительно одного веса, выполненная для всех весов). dJ/dw зависит от выбора функции потерь. Наиболее распространена функция потерь среднеквадратичной ошибки.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Рисунок 11. Функция потерь среднеквадратичной ошибки

В искусственных нейронных сетях есть важное понятие – функция активации. Функция активации (Activation function) определяет выходное

значение нейрона в зависимости от калькуляции взвешенной суммы входов и порогового значения.

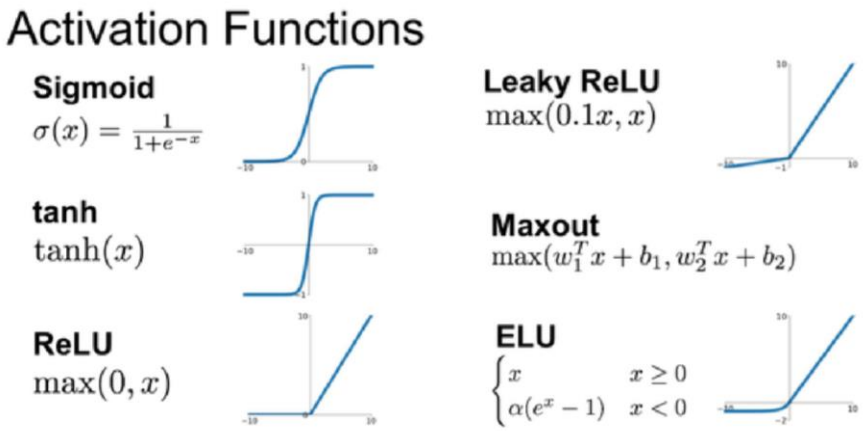

Рисунок 12. Функции активации

Каждая функция активации по-своему особенна, следовательно нельзя сказать, что есть функция активации, которая подошла бы на все задачи машинного обучения. Выбор функции активации, задача экспериментальная, т.е. надо протестировать разные виды функции активации и определить какая функция подходит для задачи и нейросеть быстро обучается.

1.3 Нейронные сети в системах управления

Системы контроля на основе Пропорционально-Интегрально-Дифференциального регулятора (ПИД-регулятор) показывает хорошие результаты настройки систем автоматического управления. Однако в сложных моделях расчет параметров по формулам не дает оптимальной настройки регулятора, поскольку аналитически полученные результаты основываются на сильно упрощенных моделях объекта. Поэтому после расчета параметров регулятора выполняется его подстройка. Подстройку можно выполнить на основе правил, которые используются для ручной настройки. Эти правила получены из опыта, теоретического анализа и численных экспериментов и связаны со свойствами элементов ПИД-регулятора [7]. Дополнительная подстройка параметров регулятора сложной системы вручную – довольно непростой процесс, так как соотношение между коэффициентами регулятора играет такую же большую роль, как и величины самих параметров. Настройка компонент ПИД-регулятора требует детального математического описания системы и представлений о том, в каких условиях он будет использоваться.

Нейроуправление (англ. Neurocontrol) — частный случай интеллектуального управления, использующий искусственные нейронные сети для решения задач управления динамическими объектами. Нейроуправление, с другой стороны, это совокупность таких дисциплин, как искусственный интеллект, нейрофизиология, теория автоматического управления и робототехника. Нейросети обладают рядом уникальных свойств,

которые делают их эффективным инструментом для создания систем контроля: способностью к обучению на примерах и обобщению данных, способностью адаптироваться к изменению свойств объекта управления и внешней среды, пригодностью для синтеза нелинейных регуляторов, высокой устойчивостью к повреждениям своих элементов в силу изначально заложенного в нейросетевую архитектуру параллелизма. Термин «нейроуправление» впервые был использован одним из авторов метода обратного распространения ошибки Полом Дж. Вербосом в 1976 году[8]. Известны многочисленные примеры практического применения нейронных сетей для решения задач управления самолётом, вертолётом, автомобилем, скоростью вращения вала двигателя, гибридным двигателем автомобиля, электропечью, турбогенератором, сварочным аппаратом, пневмоцилиндром, системой управления вооружением легкобронированных машин, моделью перевернутого маятника.

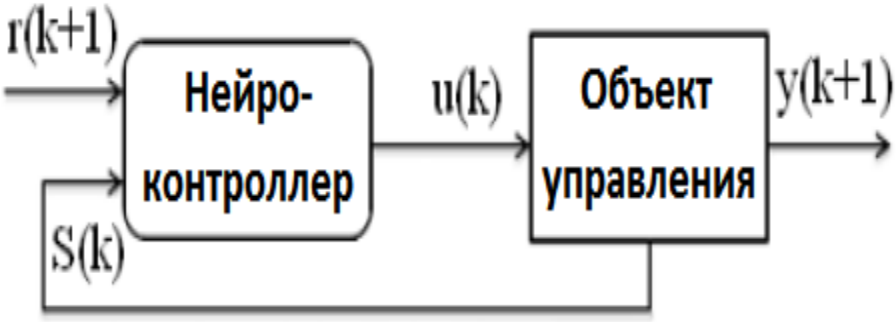

Рисунок 13. Схема прямого нейроуправления с обратной связью

По способу использования искусственных нейронных сетей методы нейроконтроля делятся на прямые и непрямые. В прямых методах нейросеть обучается генерировать управляющие сигналы на объект, в непрямых методах нейронная сеть обучается совершать вспомогательные функции: идентификация объекта управления, подавление шума, оперативная настройка коэффициентов ПИД-контроллера. В зависимости от числа нейронных сетей, составляющих нейроконтроллер, системы нейроуправления делятся на одномодульные и многомодульные. Также есть системы нейроконтроля, которые используются совместно с традиционными регуляторами, называются гибридными.[9]

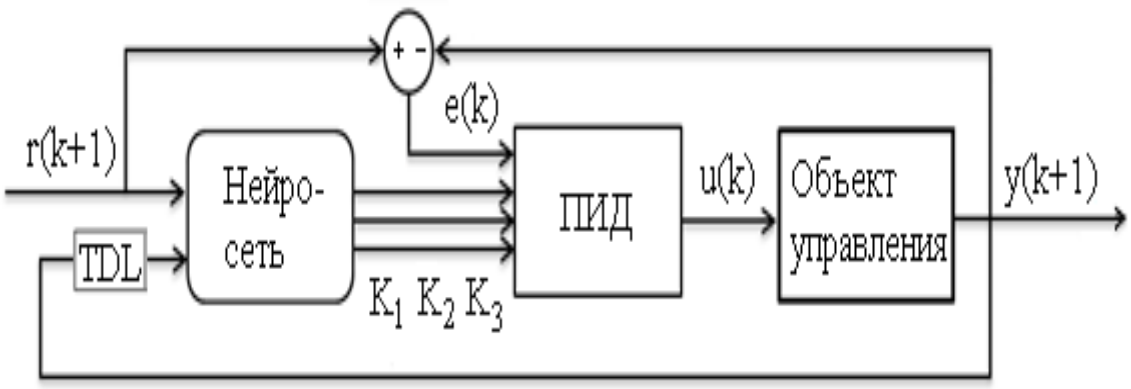

Рисунок 14. Схема гибридного нейро-ПИД управления

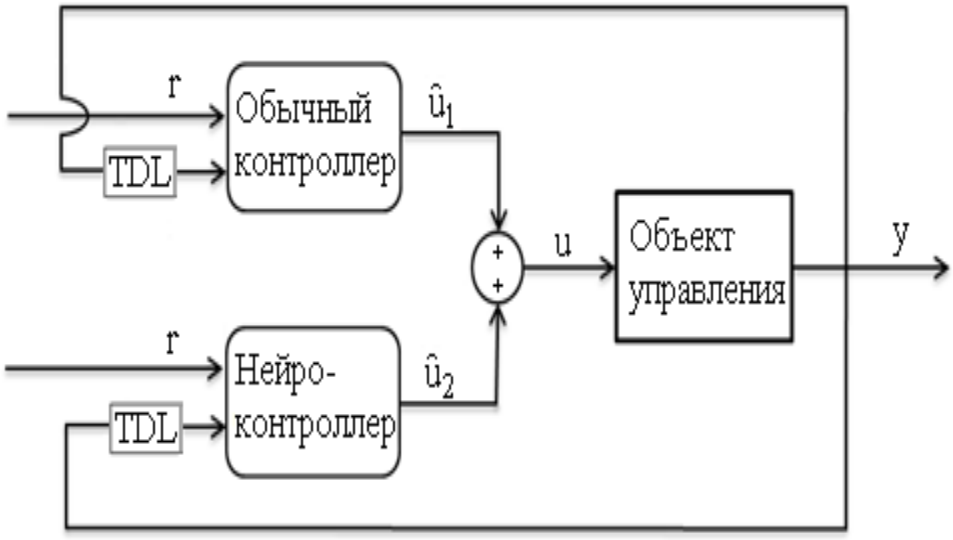

Рисунок 15. Схема гибридного параллельного нейроуправления

Метод нейроуправления с эталонной моделью (model reference adaptive control, neural adaptive control) — архитектура нейроуправления по методу обратного пропуска ошибки через прямой нейроэмулятор с дополнительно внедрённой в схему эталонной моделью (reference model) динамической системы, чтобы контролировать поведение которой обучается нейроконтроллер. Это необходимо в целях повышения качества переходного процесса: в случае, когда переход объекта в целевое положение за один такт невозможен, траектория движения и время осуществления переходного процесса становятся плохо прогнозируемыми величинами и могут привести к неустойчивости переходного процесса. Для уменьшения этой неопределенности вводится эталонная модель, представляющая собой, как правило, устойчивую линейную динамическую систему первого или второго порядка.

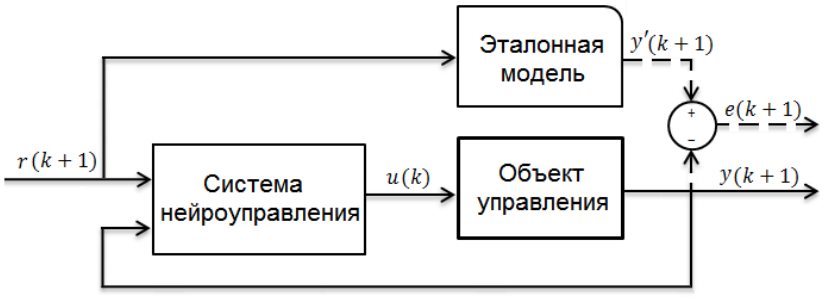

Рисунок 16. Нейроуправление с эталонной моделью

1.4 Система управления БПЛА

Основным в этой главе является рассмотрение метода управления, и промоделировать его. Акцент был сделан на контроль высоты полета, так как контроль высоты является первоочередной задачей в управлении беспилотным летательным аппаратом.

Модель (9) описывает дифференциальные уравнения системы. Рекомендуется упрощать модель системы управления с целью соответствия ограничениям в реальном времени, предъявляемых к контуру встроенной системы. Следовательно, можем пренебречь моментом вращения и силой действующей на винт, а коэффициенты тяги и лобового сопротивления примем постоянными. Система может быть переписана в форме пространства состояния X = f(X, U) , где U вектор входных сигналов и X вектор состояния который задается следующим образом:

Вектор состояния

$$X = [\varphi\ \dot{\varphi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi}\ z\ \dot{z}\ x\ \dot{x}\ \ y\dot{y}]^T, \quad (1)$$

$$\left(\begin{array}{c|c} x_1 = \varphi & x_7 = z \\ x_2 = x_1 = \dot{\varphi} & x_8 = x_7 = \dot{z} \\ x_3 = \theta & x_9 = x \\ x_4 = \dot{x_3} = \dot{\theta} & x_{10} = \dot{x_9} = \dot{x} \\ x_5 = \psi & x_{11} = y \\ x_6 = \dot{x_5} = \dot{\psi} & x_{12} = \dot{x_{11}} = \dot{y} \end{array}\right), \quad (2)$$

$$U = [U_1\ \ U_2\ \ U_3\ \ U_4]^T, \quad (3)$$

Где входные сигналы имеют следующий вид:

$$\begin{cases} U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 = b(-\Omega_2^2 + \Omega_4^2) \\ U_3 = b(\Omega_1^2 - \Omega_3^2) \\ U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{cases}, \quad (4)$$

Матрица перехода между скоростью изменения углов ориентации ($\dot{\varphi}$, $\dot{\psi}$, $\dot{\theta}$) и изменением угловых скоростей тела (p,q.r) можно рассматривать как единичную матрицу, если возмущение от висения невелики.

$$f(X,U) = \begin{pmatrix} \dot{\varphi} \\ \dot{\theta}\dot{\psi}a_1 + \dot{\theta}a_2\Omega_r + b_1U_2 \\ \dot{\theta} \\ \dot{\varphi}\dot{\psi}a_3 + \dot{\varphi}a_4\Omega_r + b_2U_3 \\ \dot{\psi} \\ \dot{\theta}\dot{\varphi}a_5 + b_3U_4 \\ \dot{z} \\ g - (\cos\varphi\cos\theta)\frac{1}{m}U_1 \\ \dot{x} \\ u_x\frac{1}{m}U_1 \\ \dot{y} \\ u_y\frac{1}{m}U_1 \end{pmatrix}, \quad (5)$$

Где:

$$\begin{array}{l|l} a_1 = (I_{yy} - I_{zz})/I_{xx} & b_1 = l/I_{xx} \\ a_2 = J_r/I_{xx} & b_2 = l/I_{yy} \\ a_3 = (I_{zz} - I_{xx})/I_{yy} & b_3 = 1/I_{zz} \\ a_4 = J_r/I_{yy} & \\ a_5 = (I_{xx} - I_{yy})/I_{zz} & \end{array}, \quad (6)$$

$$u_x = (\cos\varphi\sin\theta\cos\psi + \sin\varphi\sin\psi)$$

$$u_y = (cos\,\varphi\,sin\,\theta\,sin\,\psi - sin\,\varphi\,cos\,\psi), \quad (7)$$

Стоит отметить, что углы и их временные производные не зависят от преобразующих компонентов. С другой стороны, переходная система зависит от углов.

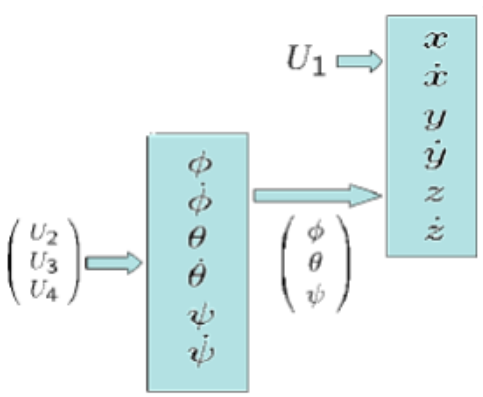

Рисунок 17 – Соединение подсистемы углов поворота и линейного перемещения.

1.5 Управление с использованием ПИД-регулятора

ПИД - регулятор был изобретен ещё в 1910 году; позднее, в 1942 г., Зиглер и Никольс разработали методику настройки ПИД - регулятора, а появления микропроцессоров в 80-х годах, поспособствовало развитию ПИД - регуляторов. ПИД - регулятор относится к самому распространенному типу регуляторов. около 90-95% регуляторов, находящихся в настоящее время в эксплуатации, основываются на ПИД алгоритме. Причиной столь высокой популярности является простота построения и промышленного применения, понятность функционирования, пригодность для решения большинства практических задач и низкая стоимость.

Для того, чтобы сделать возможным создание нескольких ПИД - регуляторов для исследуемой системы, можно пренебречь этими гироскопическими эффектами и тем самым устранить перекрестную связь в системе управления.

$$\begin{cases} I_{xx}\ddot{\varphi} = lU_2 \\ I_{yy}\ddot{\theta} = lU_3 \\ I_{zz}\ddot{\psi} = U_4 \end{cases}, \quad (8)$$

Если в (8) добавить, динамику роторов и перезаписать модель в Лапласовом изображении, то получим

$$\begin{cases} \varphi(s) = \dfrac{B^2 bl}{s^2(s+A)^2 I_{xx}}(u_4^2(s) - u_2^2(s)) \\ \theta(s) = \dfrac{B^2 bl}{s^2(s+A)^2 I_{yy}}(u_3^2(s) - u_1^2(s)) \\ \psi(s) = \dfrac{B^2 d}{s^2(s+A)^2 I_{zz}}(-1)^{i+1}\sum_{i=1}^{4} u_i^2(s) \end{cases}, \quad (9)$$

Где A и B это коэффициенты линеаризованной с помощью ряда Тэйлора динамики ротора, тогда как коэффициент C, слишком мал по сравнению с B, и им можно пренебречь. Используя управляющие воздействия вместо входных сигналов моторов, получаем:

$$\begin{cases} \varphi(s) = \dfrac{B^2 bl}{s^2(s+A)^2 I_{xx}} U_2 \\ \theta(s) = \dfrac{B^2 bl}{s^2(s+A)^2 I_{yy}} U_3 \\ \psi(s) = \dfrac{B^2 d}{s^2(s+A)^2 I_{zz}} U_4 \end{cases}, \quad (10)$$

1.6 Реализация модели системы управления квадрокоптером и примеры работы

Исходя из изученного теоретического материала, необходимо перейти к практической реализации. А именно к построению модели квадрокоптера в математическом пакете MatLab Simulink. MATLAB — это высокоуровневый язык и интерактивная среда для программирования, численных расчетов и визуализации результатов. С помощью MATLAB можно анализировать данные, разрабатывать алгоритмы, создавать модели и приложения. Simulink – это графическая среда имитационного моделирования, позволяющая при помощи блок-диаграмм в виде направленных графов, строить динамические модели, включая дискретные, непрерывные и гибридные, нелинейные и разрывные системы. [10]

В ходе создания модели системы управления квадрокоптером, она была разбита на отдельные блоки, чтобы упростить дальнейшее ее использование в исследованиях:

а) блок задания начальных условий;

б) блок контроллера управления высотой;

в) блок управления движением;

г) блок описывающий динамику квадрокоптера.

Разработанная система управления представлена на рисунке 9. Представленная система состоит из 4 модулей. Каждый модуль это часть общей системы управления. Данные модули являются активными элементами и при двойном нажатии на него, открывается структура этого блока. Такая методика создания системы управления позволяет более легко взаимодействовать с ней, путем быстрого доступа к определенной части системы и при необходимости, быстро изменить ее. Так же на схеме присутствует две активные запрограммированные кнопки. Система имеет принцип управление по отклонению, которая организована путем добавления отрицательной обратной связи. Управляющее воздействие при использовании принципа управления по отклонению вырабатывается в результате

преобразования отклонения управляемой величины от требуемого значения. Далее будет разобран каждый блок по отдельности.

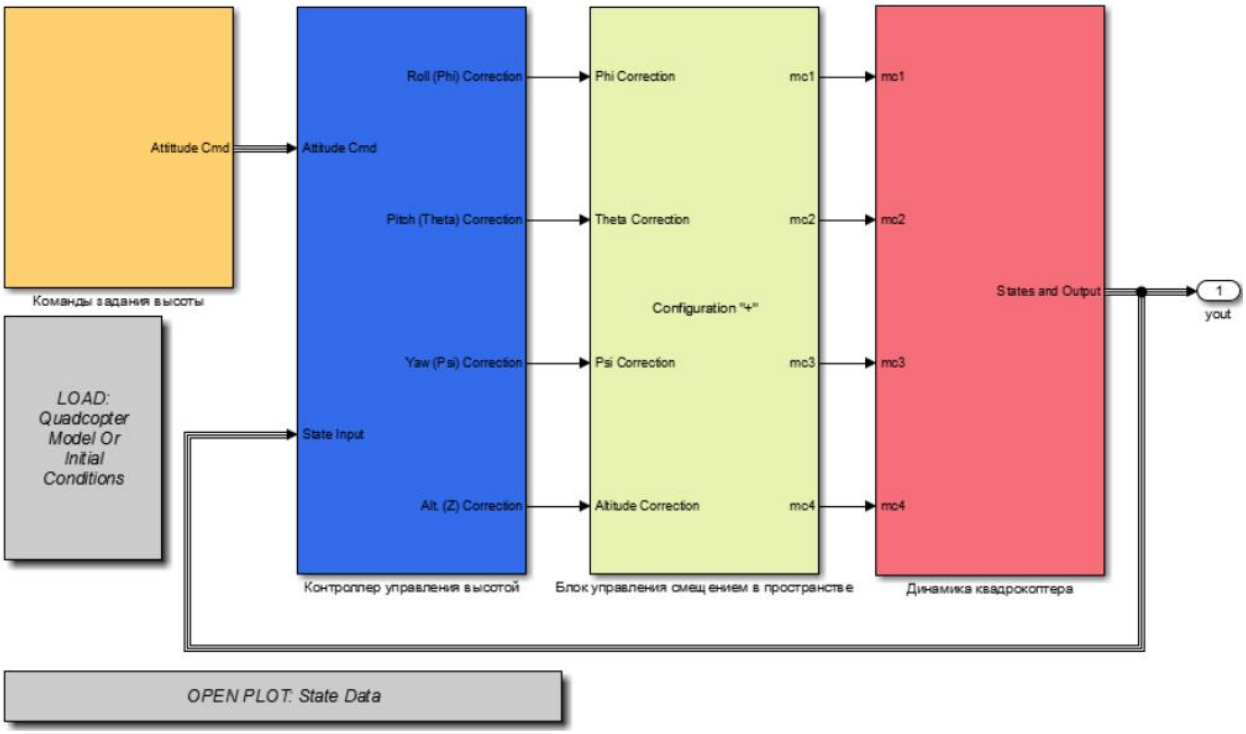

Рисунок 18 – Модель системы управления БПЛА

2 Языки программирования и фреймворки для алгоритмов машинного обучения

2.1 Python, TensorFlow, PyTorch

Python считается самым простым языком программирования — именно поэтому он самый распространенный язык. Кроме простоты, язык Python может довольно легко взаимодействовать с другими языками, особенно с C и C++.

Язык Python лучше всего подходит для выполнения задач машинного обучения, потому что этот язык довольно понятный по сравнению с другими языками. Более того, у языка Python отличная производительность при обработке больших объемов данных.

Одна из основных причин, почему Python используется для машинного обучения состоит в том, что у него есть множество фреймворков, которые

упрощают процесс написания кода и сокращают время на разработку. Все комплексные математические функции обработки данных заложены во фреймворке, после чего разработка алгоритмов машинного обучения становится простым, быстрым и модульным.

Простой синтаксис языка Python позволяет программисту тестировать комплексные алгоритмы с минимальной тратой времени на их реализацию.

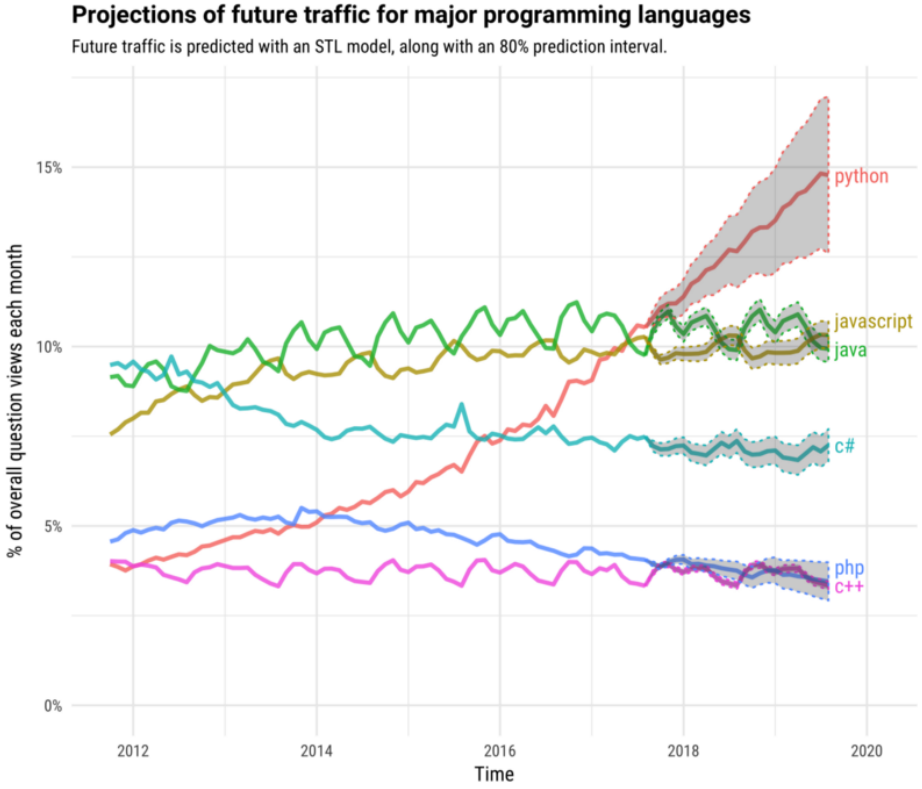

Рисунок 19. Популярность языка Python

Компания Google активно использует собственный фреймворк глубокого обучения – TensorFlow для таких крупномасштабных сервисов как почтовый сервис Gmail и Google Translate. TensorFlow уже применяют такие значительные компании как Uber, Airbnb, Dropbox и многие другие. Данный фреймворк оказал огромное влияние на разработку алгоритмов машинного обучения и оптимизации бизнес-процессов.

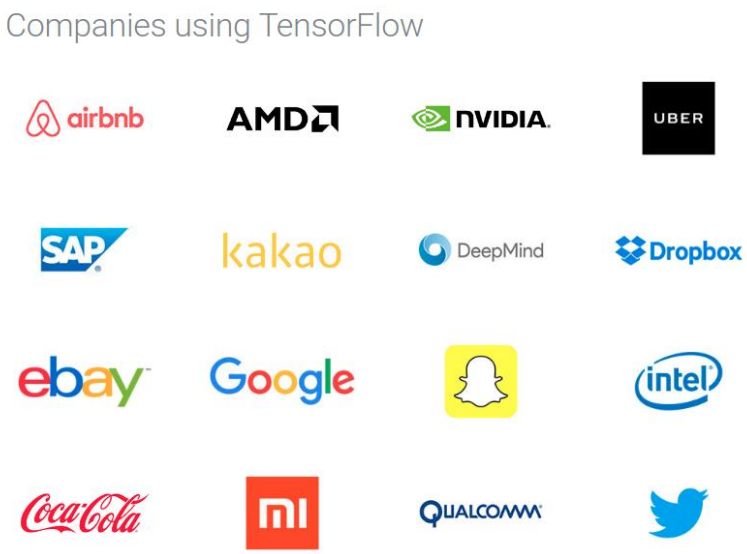

Рисунок 20. Компании, использующие TensorFlow

Наиболее комфортным и простым клиентским языком работы с TensorFlow является язык Python, но доступны и другие интерфейсы на языках JavaScript, C++, Java и Go. Open source сообщество разработало также решения для яызков C# и Julia.

Фреймворк PyTorch был разработан компанией Facebook, но уже используется для собственных задач такими компаниями как Twitter и Salesforce. На данный момент этот фреймворк также популярен, есть большое сообщество разработчиков и есть открытая документация. В отличие от TensorFlow, библиотека PyTorch оперирует динамически обновляемым графом. То есть позволяет вносить изменения в архитектуру в процессе работы.

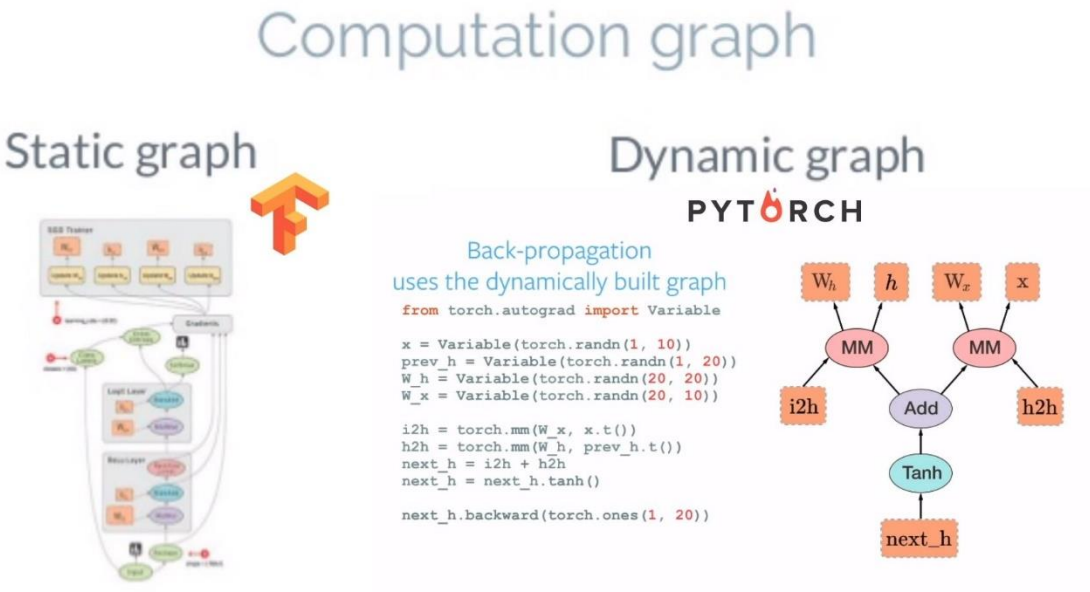

Рисунок 21. Сравнение графов TensorFlow и PyTorch

2.2 C#, OpenNN

OpenNN (Open Neural Networks Library) — это библиотека программного обеспечения, написанная на языке программирования C++, которая дает разрабатывать нейронные сети, основная область исследований в области глубокого обучения. Библиотека с открытым исходным кодом лицензируется в соответствии с GNU Lesser General Public License.

Разработка открытой библиотеки OpenNN нейронных сетей началась в 2003 году в Международном центре вычислительных методов в машиностроении (CIMNE) в рамках исследовательского проекта и имела название FLOOD, что в переводе означает наводнение. Сегодня разработками занимается компания Artelnics, специализирующаяся полностью на искусственном интеллекте.

2.3 Matlab

MATLAB (сокращение от англ. «Matrix Laboratory», в русском языке произносится как Матлаб) — пакет прикладных программ для решения задач технических вычислений. Пакет используют более миллиона инженерных и научных работников, он работает на большинстве современных операционных систем, включая Linux, Mac OS и MS Windows. Также в пакет Matlab входит нейросетевые пакеты и различные инструменты обработки данных и математической статистики.

Язык MATLAB является высокоуровневым интерпретируемым языком программирования, включающим основанные на матрицах структуры данных, широкий спектр функций, интегрированную среду разработки, объектно-ориентированные возможности и интерфейсы к программам, написанным на других языках программирования. Программы, написанные на MATLAB, бывают двух типов — функции и скрипты. Язык Matlab прост, и местами похож на язык Python. Функции имеют input и output аргументы, а также собственное рабочее пространство для хранения промежуточных результатов вычислений и переменных. Скрипты же используют общее рабочее пространство. Как скрипты, так и функции сохраняются в виде текстовых файлов и компилируются в машинный код динамически. Существует также возможность сохранять так называемые pre-parsed программы — функции и скрипты, обработанные в вид, удобный для машинного исполнения. В общем случае такие программы выполняются быстрее обычных, особенно если функция содержит команды построения графиков.

3 Классификация радиолокационных целей с использованием машинного обучения и глубокого обучения

В данной работе показано, как классифицировать радиолокационную отдачу с помощью машинного и глубокого обучения. Подход машинного обучения использует извлечение признаков вейвлет-рассеяния в сочетании с методом опорных векторов. Кроме того, проиллюстрированы два подхода глубокого обучения: трансферное обучение с использованием SqueezeNet и рекуррентная нейронная сеть с длительной кратковременной памятью (LSTM).

3.1 Генерирование синтетических паттернов

Здесь показано, как создавать синтезированные данные для алгоритма обучения. Следующий код имитирует схему RCS цилиндра с радиусом 1 метр и высотой 10 метров. Рабочая частота радара составляет 850 МГц.

```
c = 3e8;
fc = 850e6;
[cylrcs,az,el] = rcscylinder(1,1,10,c,fc);
helperTargetRCSPatternPlot(az,el,cylrcs);
```

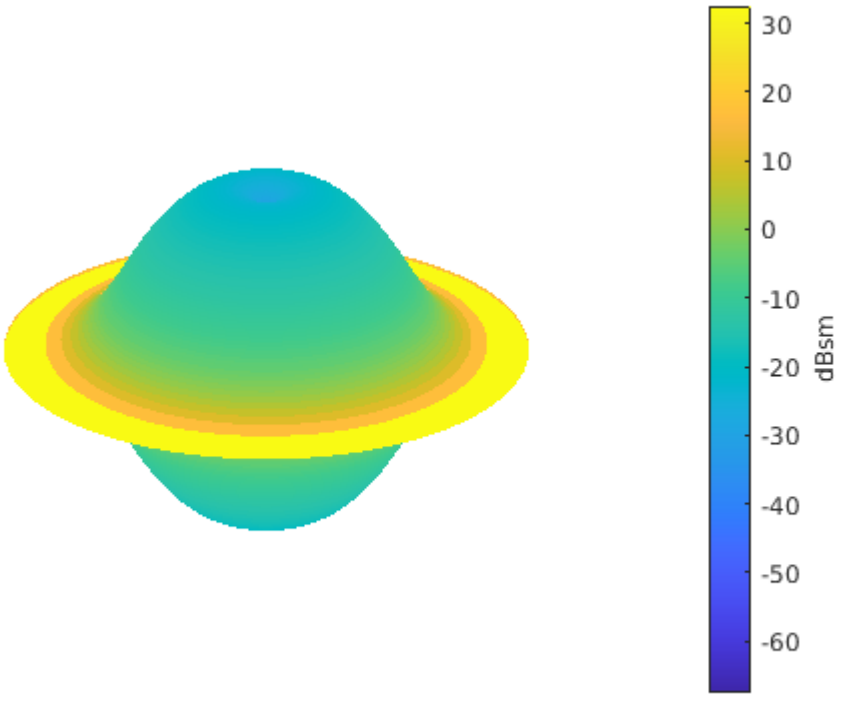

Рисунок 22. Паттерн цилиндра

На следующем графике показано, как моделировать 100 возвратов цилиндра с течением времени. Предполагается, что под цилиндром

происходит движение, которое вызывает небольшие вибрации вокруг прицела, в результате чего угол наклона изменяется от одного образца к другому.

```
rng default;
N = 100;
az = 2*randn(1,N);
el = 2*randn(1,N);
cylrtn = cyltgt(ones(1,N),[az;el]);


plot(mag2db(abs(cylrtn)));
xlabel('Time Index')
ylabel('Target Return (dB)');
title('Target Return for Cylinder');
```

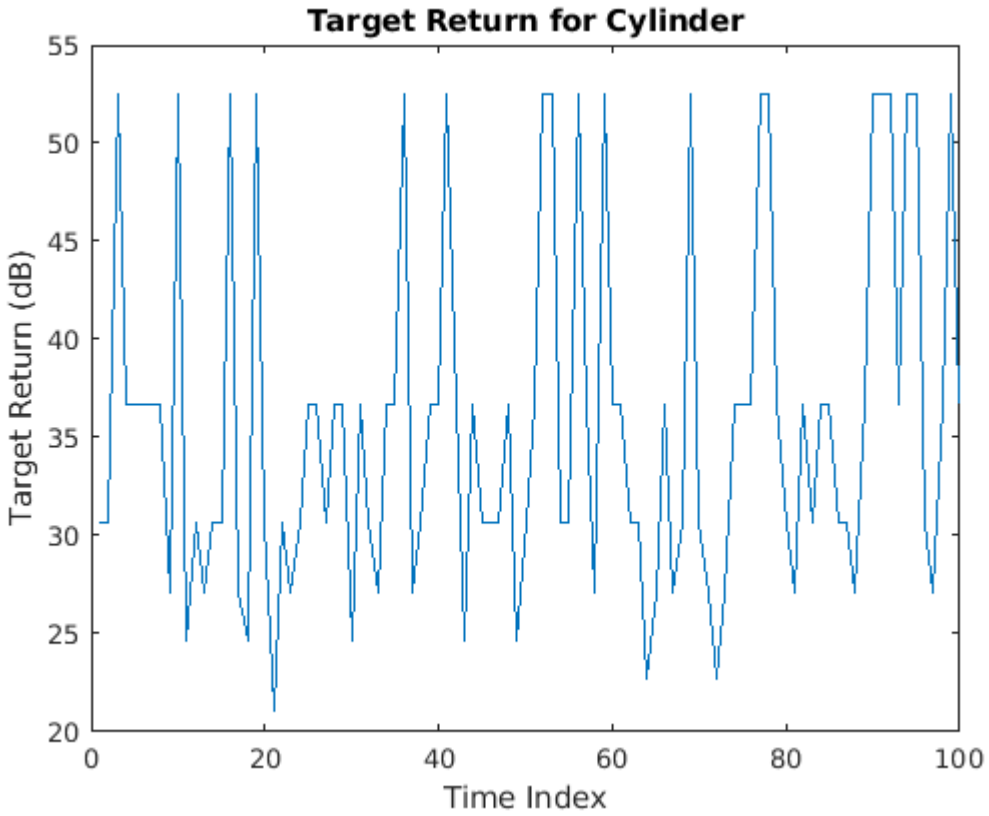

Рисунок 23. Целевые значения для цилиндра

Возвращение конуса можно сгенерировать аналогично. Для создания обучающего набора описанный выше процесс повторяется для 5 произвольно выбранных радиусов цилиндра. Кроме того, для каждого радиуса 10 профилей движения моделируются путем изменения угла падения после 10 случайно сформированных синусоидальных кривых вокруг точки прицеливания. В каждом профиле движения 701 выборка, поэтому выборка 701 на 50. Процесс повторяется для мишени для цилиндра, в результате чего получается матрица

тренировок 701 на 100 с 50 профилями цилиндров и 50 конусов. В тестовом наборе мы используем 25 цилиндрических и 25 конических профилей для создания тренировочного набора 701 на 50. Из-за длительного времени вычислений данные обучения предварительно вычисляются и загружаются ниже.

```
subplot(2,2,1)
plot(cylinderAspectAngle(1,:))
ylim([-90 90])
grid on
title('Cylinder Aspect Angle vs. Time'); xlabel('Time Index'); ylabel('Aspect Angle (degrees)');
subplot(2,2,3)
plot(RCSReturns.Cylinder_1); ylim([-50 50]);
grid on
title('Cylinder Return'); xlabel('Time Index'); ylabel('Target Return (dB)');
subplot(2,2,2)
plot(coneAspectAngle(1,:)); ylim([-90 90]); grid on;
title('Cone Aspect Angle vs. Time'); xlabel('Time Index'); ylabel('Aspect Angle (degrees)');
subplot(2,2,4);
plot(RCSReturns.Cone_1); ylim([-50 50]); grid on;
title('Cone Return'); xlabel('Time Index'); ylabel('Target Return (dB)');
```

3.2 Вейвлет рассеивание

В экстракторе признаков вейвлет-рассеяния данные распространяются через серию вейвлет-преобразований, нелинейностей и усреднения, чтобы получить представления с малыми дисперсиями временных рядов. Вейвлет-рассеяние во времени дает представления сигнала, нечувствительные к сдвигам во входном сигнале, не жертвуя различимостью класса. Ключевыми параметрами, которые необходимо указать в разложении вейвлет-рассеяния времени, являются масштаб инварианта времени, число вейвлет-преобразований и количество вейвлетов на октаву в каждом из банков вейвлет-фильтров. Во многих приложениях каскад из двух банков фильтров достаточен для достижения хорошей производительности. В этом примере мы строим декомпозицию вейвлет-рассеяния во времени с банками фильтров по умолчанию: 8 вейвлетов на октаву в первом банке фильтров и 1 вейвлет на октаву во втором банке фильтров. Шкала инвариантности установлена в 701 отсчете, длина данных.

```
sf = waveletScattering('SignalLength',701,'InvarianceScale',701);
sTrain = sf.featureMatrix(RCSReturns{:,:},'transform','log');
sTest = sf.featureMatrix(RCSReturnsTest{:,:},'transform','log');
TrainFeatures = squeeze(mean(sTrain,2))';
```

```
TestFeatures = squeeze(mean(sTest,2))';
TrainLabels = repelem(categorical({'Cylinder','Cone'}),[50 50])';
TestLabels = repelem(categorical({'Cylinder','Cone'}),[25 25])';
```

3.3 Обучения модели

Обучение алгоритма метода опорных векторов с квадратичным ядром для характеристик рассеяния и получение точности перекрестной проверки.

```
template = templateSVM('KernelFunction', 'polynomial', ...
   'PolynomialOrder', 2, ...
   'KernelScale', 'auto', ...
   'BoxConstraint', 1, ...
   'Standardize', true);
classificationSVM = fitcecoc(...
   TrainFeatures, ...
   TrainLabels, ...
   'Learners', template, ...
   'Coding', 'onevsone', ...
   'ClassNames', categorical({'Cylinder','Cone'}));
partitionedModel = crossval(classificationSVM, 'KFold', 5);
[validationPredictions, validationScores] = kfoldPredict(partitionedModel);
validationAccuracy = (1 - kfoldLoss(partitionedModel, 'LossFun',
'ClassifError'))*100
```

3.4 Классификация

Используя обученный SVM, классификация характеристики рассеяния, полученные из тестового набора.

```
predLabels = predict(classificationSVM,TestFeatures);
accuracy = sum(predLabels == TestLabels )/numel(TestLabels)*100
figure('Units','normalized','Position',[0.2 0.2 0.5 0.5]);
ccDCNN = confusionchart(TestLabels,predLabels);
ccDCNN.Title = 'Confusion Chart';
ccDCNN.ColumnSummary = 'column-normalized';
ccDCNN.RowSummary = 'row-normalized';
```

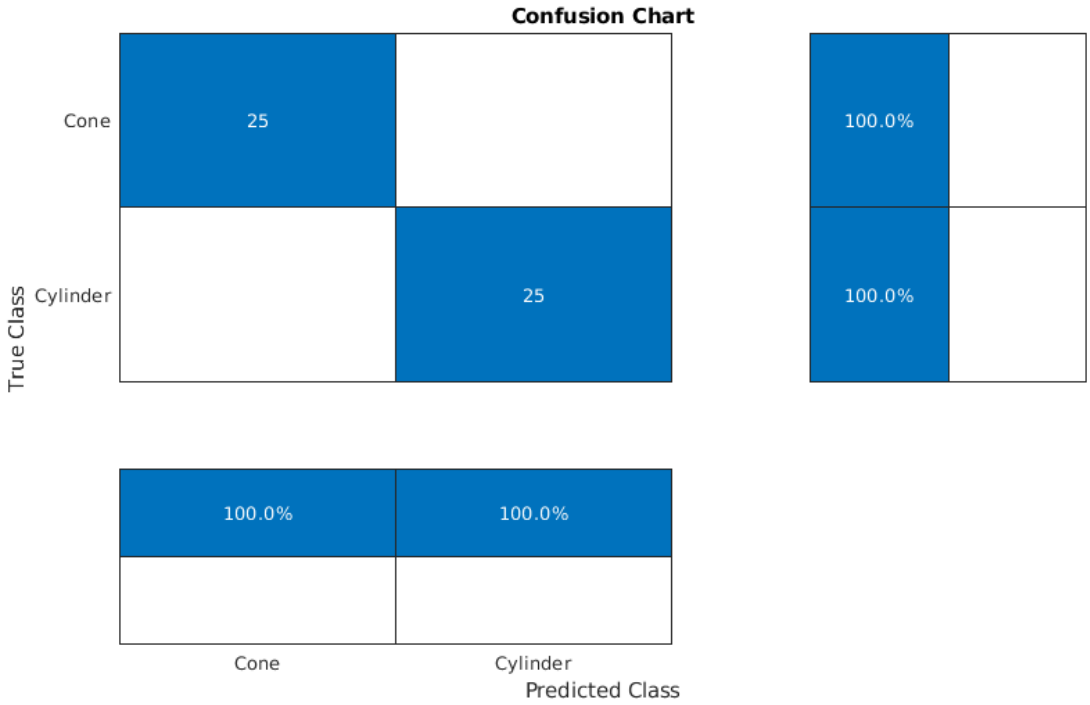

Рисунок 24. График неопределённости

3.5 Использование сверточных нейронных сетей

SqueezeNet - это глубокая сверточная нейронная сеть (CNN), подготовленная для изображений в 1000 классах, как это используется в конкурсе визуального распознавания изображений ImageNet (ILSVRC). В этом примере мы повторно используем предварительно обученную SqueezeNet для классификации радиолокационных сигналов, принадлежащих одному из двух классов.

```
ans =
  68x1 Layer array with layers:

     1   'data'                    Image Input           227x227x3 images with
'zerocenter' normalization
     2   'conv1'                   Convolution           64 3x3x3 convolutions with
stride [2  2] and padding [0  0  0  0]
     3   'relu_conv1'              ReLU                  ReLU
     4   'pool1'                   Max Pooling           3x3 max pooling with stride
[2  2] and padding [0  0  0  0]
     5   'fire2-squeeze1x1'        Convolution           16 1x1x64 convolutions
with stride [1  1] and padding [0  0  0  0]
     6   'fire2-relu_squeeze1x1'   ReLU                  ReLU
     7   'fire2-expand1x1'         Convolution           64 1x1x16 convolutions
with stride [1  1] and padding [0  0  0  0]
```

8 'fire2-relu_expand1x1' ReLU ReLU
9 'fire2-expand3x3' Convolution 64 3x3x16 convolutions with stride [1 1] and padding [1 1 1 1]
10 'fire2-relu_expand3x3' ReLU ReLU
11 'fire2-concat' Depth concatenation Depth concatenation of 2 inputs
12 'fire3-squeeze1x1' Convolution 16 1x1x128 convolutions with stride [1 1] and padding [0 0 0 0]
13 'fire3-relu_squeeze1x1' ReLU ReLU
14 'fire3-expand1x1' Convolution 64 1x1x16 convolutions with stride [1 1] and padding [0 0 0 0]
15 'fire3-relu_expand1x1' ReLU ReLU
16 'fire3-expand3x3' Convolution 64 3x3x16 convolutions with stride [1 1] and padding [1 1 1 1]
17 'fire3-relu_expand3x3' ReLU ReLU
18 'fire3-concat' Depth concatenation Depth concatenation of 2 inputs
19 'pool3' Max Pooling 3x3 max pooling with stride [2 2] and padding [0 1 0 1]
20 'fire4-squeeze1x1' Convolution 32 1x1x128 convolutions with stride [1 1] and padding [0 0 0 0]
21 'fire4-relu_squeeze1x1' ReLU ReLU
22 'fire4-expand1x1' Convolution 128 1x1x32 convolutions with stride [1 1] and padding [0 0 0 0]
23 'fire4-relu_expand1x1' ReLU ReLU
24 'fire4-expand3x3' Convolution 128 3x3x32 convolutions with stride [1 1] and padding [1 1 1 1]
25 'fire4-relu_expand3x3' ReLU ReLU
26 'fire4-concat' Depth concatenation Depth concatenation of 2 inputs
27 'fire5-squeeze1x1' Convolution 32 1x1x256 convolutions with stride [1 1] and padding [0 0 0 0]
28 'fire5-relu_squeeze1x1' ReLU ReLU
29 'fire5-expand1x1' Convolution 128 1x1x32 convolutions with stride [1 1] and padding [0 0 0 0]
30 'fire5-relu_expand1x1' ReLU ReLU
31 'fire5-expand3x3' Convolution 128 3x3x32 convolutions with stride [1 1] and padding [1 1 1 1]
32 'fire5-relu_expand3x3' ReLU ReLU
33 'fire5-concat' Depth concatenation Depth concatenation of 2 inputs
34 'pool5' Max Pooling 3x3 max pooling with stride [2 2] and padding [0 1 0 1]
35 'fire6-squeeze1x1' Convolution 48 1x1x256 convolutions with stride [1 1] and padding [0 0 0 0]

36 'fire6-relu_squeeze1x1' ReLU ReLU
37 'fire6-expand1x1' Convolution 192 1x1x48 convolutions with stride [1 1] and padding [0 0 0 0]
38 'fire6-relu_expand1x1' ReLU ReLU
39 'fire6-expand3x3' Convolution 192 3x3x48 convolutions with stride [1 1] and padding [1 1 1 1]
40 'fire6-relu_expand3x3' ReLU ReLU
41 'fire6-concat' Depth concatenation Depth concatenation of 2 inputs
42 'fire7-squeeze1x1' Convolution 48 1x1x384 convolutions with stride [1 1] and padding [0 0 0 0]
43 'fire7-relu_squeeze1x1' ReLU ReLU
44 'fire7-expand1x1' Convolution 192 1x1x48 convolutions with stride [1 1] and padding [0 0 0 0]
45 'fire7-relu_expand1x1' ReLU ReLU
46 'fire7-expand3x3' Convolution 192 3x3x48 convolutions with stride [1 1] and padding [1 1 1 1]
47 'fire7-relu_expand3x3' ReLU ReLU
48 'fire7-concat' Depth concatenation Depth concatenation of 2 inputs
49 'fire8-squeeze1x1' Convolution 64 1x1x384 convolutions with stride [1 1] and padding [0 0 0 0]
50 'fire8-relu_squeeze1x1' ReLU ReLU
51 'fire8-expand1x1' Convolution 256 1x1x64 convolutions with stride [1 1] and padding [0 0 0 0]
52 'fire8-relu_expand1x1' ReLU ReLU
53 'fire8-expand3x3' Convolution 256 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]
54 'fire8-relu_expand3x3' ReLU ReLU
55 'fire8-concat' Depth concatenation Depth concatenation of 2 inputs
56 'fire9-squeeze1x1' Convolution 64 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]
57 'fire9-relu_squeeze1x1' ReLU ReLU
58 'fire9-expand1x1' Convolution 256 1x1x64 convolutions with stride [1 1] and padding [0 0 0 0]
59 'fire9-relu_expand1x1' ReLU ReLU
60 'fire9-expand3x3' Convolution 256 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]
61 'fire9-relu_expand3x3' ReLU ReLU
62 'fire9-concat' Depth concatenation Depth concatenation of 2 inputs
63 'drop9' Dropout 50% dropout
64 'conv10' Convolution 1000 1x1x512 convolutions with stride [1 1] and padding [0 0 0 0]

65 'relu_conv10' ReLU ReLU
66 'pool10' Global Average Pooling Global average pooling
67 'prob' Softmax softmax
68 'ClassificationLayer_predictions' Classification Output crossentropyex with 'tench' and 999 other classes

3.6 Вейвлет преобразование

SqueezeNet предназначен для распознавания различий в изображениях и классификации результатов. Поэтому, чтобы использовать SqueezeNet для классификации радиолокационных сигналов, необходимо преобразовать временные ряды одномерного радиолокационного сигнала в изображение. Обычный способ сделать это - использовать частотно-временное представление (TFR). Существует несколько вариантов частотно-временного представления сигнала, и наиболее подходящий из них зависит от характеристик сигнала. Чтобы определить, какой TFR может подойти для этой проблемы, случайным образом выберите и нанесите несколько радиолокационных сигналов от каждого класса.

```
rng default;
idxCylinder = randperm(50,2);
idxCone = randperm(50,2)+50;
```

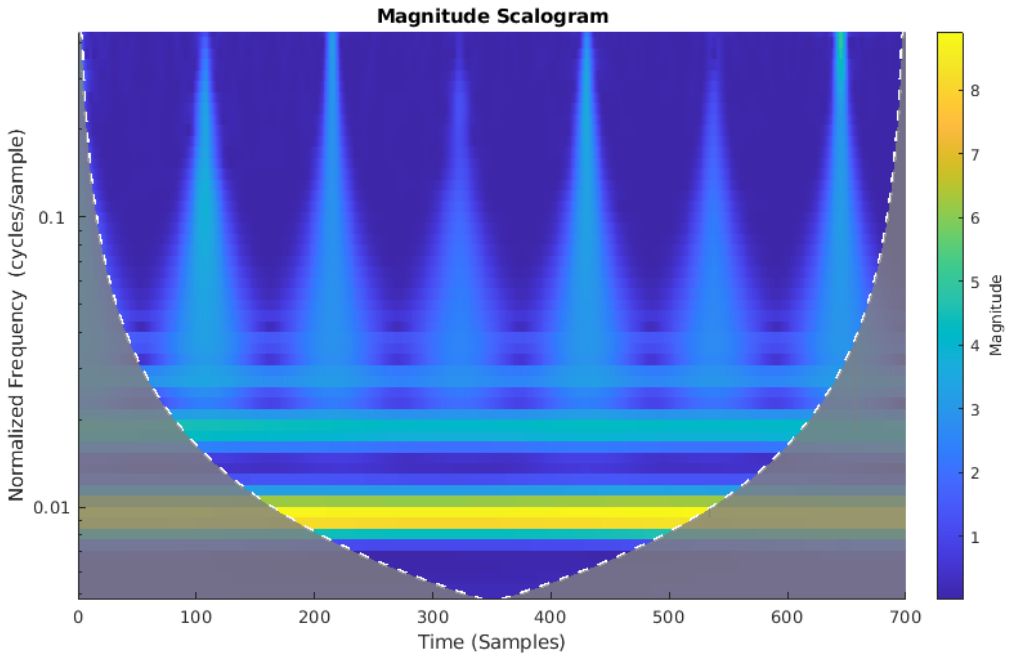

Рисунок 25. Скейлограмма

Очевидно, что показанные ранее радиолокационные сигналы характеризуются медленными переменными изменениями, перемежающимися большими переходными процессами, как описано ранее. Вейвлет-преобразование идеально подходит для разреженного представления таких сигналов. Вейвлеты сжимаются для локализации переходных явлений с высоким временным разрешением и растягиваются для захвата медленно меняющейся структуры сигнала. Получить и построить график непрерывного вейвлет-преобразования одного из возвратов цилиндров.

3.7 Подготовка изображений

Вспомогательная функция helpergenWaveletTFImg получает CWT для каждого возврата радара, изменяет форму CWT для обеспечения совместимости с SqueezeNet и записывает CWT в виде файла JPEG. Чтобы запустить helpergenWaveletTFImg, необходимо выбрать parentDir, где есть разрешение на запись. В этом примере используется tempdir, но можно использовать любую папку на вашем компьютере, где есть разрешение на запись. Вспомогательная функция создает папки обучающих и тестовых наборов в parentDir, а также создает подпапки Cylinder и Cone в обучающих и тестовых папках. Эти папки заполнены изображениями в формате JPEG, которые будут использоваться в качестве входных данных для SqueezeNet.

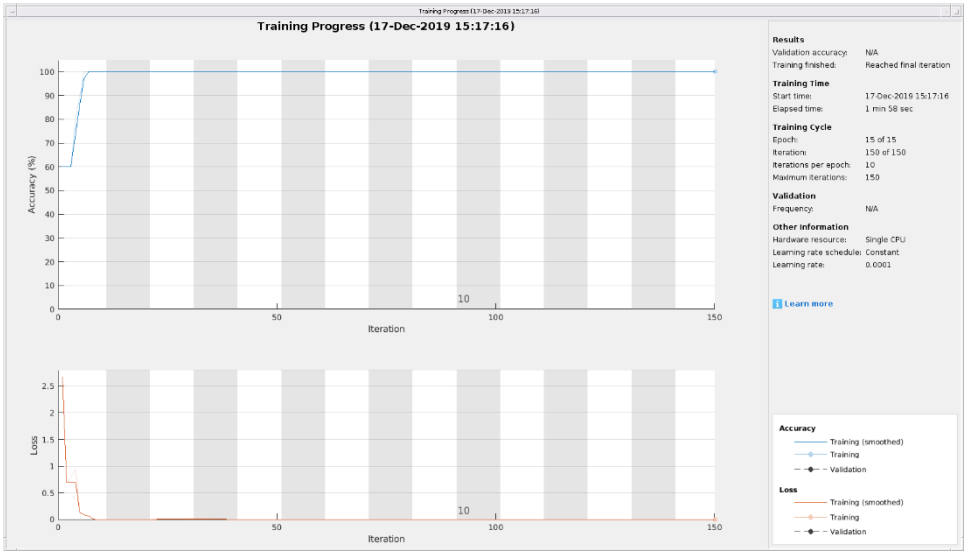

Рисунок 26. Процесс тестирования

3.8 LSTM

В последнем разделе этого примера описан рабочий процесс LSTM. Сначала определяются уровни LSTM:

```
LSTMlayers = [ ...
   sequenceInputLayer(1)
   bilstmLayer(100,'OutputMode','last')
   fullyConnectedLayer(2)
   softmaxLayer
   classificationLayer
   ];
options = trainingOptions('adam', ...
   'MaxEpochs',30, ...
   'MiniBatchSize', 150, ...
   'InitialLearnRate', 0.01, ...
   'GradientThreshold', 1, ...
   'plots','training-progress', ...
   'Verbose',false,'ExecutionEnvironment','cpu');
trainLabels = repelem(categorical({'cylinder','cone'}),[50 50]);
trainLabels = trainLabels(:);
trainData = num2cell(table2array(RCSReturns)',2);
testData = num2cell(table2array(RCSReturnsTest)',2);
testLabels = repelem(categorical({'cylinder','cone'}),[25 25]);
testLabels = testLabels(:);
RNNnet = trainNetwork(trainData,trainLabels,LSTMlayers,options);
```

Заключение

Диссертационная работа связана с исследованием методов и алгоритмов машинного обучения для нейроуправления, а также практическое использование алгоритмов машинного обучения в радиолокации. Использование алгоритмов ИИ в радиолокации имеет важное военное значение.

В данной работе исследованы языки программирования а также фреймворки и библиотеки для разработки алгоритмов искусственного интеллекта, а именно алгоритмы машинного и глубокого обучения. Рассмотретны различные архитектуры нейросетей, а также метод обратного распространения ошибки и градиентный спуск.

В данной работе представлен рабочий процесс для выполнения классификации радиолокационных целей с использованием методов машинного и глубокого обучения. Из-за характеристик сигнала вейвлет-методы использовались как для машинного обучения, так и для подходов CNN. С этим набором данных мы также получили такую же точность, просто подавая необработанные данные в LSTM.

Список использованной литературы

1 Л. Шапиро, Дж. Стокман. Компьютерное зрение = Computer Vision. — М.: Бином. Лаборатория знаний, 2006. — 752 с. — ISBN 5-94774-384-1.

2 АЛЕКСЕЙ ПОТАПОВ, СИСТЕМЫ КОМПЬЮТЕРНОГО ЗРЕНИЯ: СОВРЕМЕННЫЕ ЗАДАЧИ И МЕТОДЫ. Журнал: CONTROL ENGINEERING РОССИЯ, 1 (49), 2014.

3 Hiroyuki Arai, Kazuyuki Iso, Akira Kojima, Hitoshi Nakazawa, and Hideki Koike. Toward Intelligent Video Surveillance. NTT Technical Review, Nov. 2007, Vol. 5, No. 11.

4 Фазылова А., Тультаев Б., Балбаев Г., Султан А. Патент Казахстана №24242 от 04.07.2019 «Промышленный прибор с функцией технического зрения для доступа к шкафу управления»

5 Kyle Field. Tesla Achieved The Accuracy Of Lidar With Its Advanced Computer Vision Tech. Journal Of ClaenTechnica, April 24th, 2020

6 Сайт The Verge URL: [https://www.theverge.com/2016/3/16/11242578/movidius-myriad-2-chip-computer-vision-dji-phantom-4/The revolutionary chipmaker behind Google's project Tango is now powering DJI's autonomous drone] (дата обращения:05.12.2018)

7 Сайт Avrspot URL: [https://www.avrspot.com/using-computer-vision-to-improve-healthcare/ Using Computer Vision To Improve Healthcare] (дата обращения: 14.12.2018)

8 Официальный сайт ООН URL: [www.un.org › sections › issues-depth › population / Народонаселение/Организация Объединенных Наций] (дата обращения:20.01.2019)

9 M. A. Akhloufi "3D vision system for intelligent milking robot automation", Proc. SPIE 9025, Intelligent Robots and Computer Vision XXXI: Algorithms and Techniques, 90250N, 3 February 2014

10 J.Hemming[a]T.Rath[b] PA—Precision Agriculture: Computer-Vision-based Weed Identification under Field Conditions using Controlled Lighting Journal of Agricultural Engineering Research Volume 78, Issue 3, March 2001, Pages 233-243

Приложение А

Пример кода на Python для градиентного спуска

```
def train(X, y, W, B, alpha, max_iters):
'''
Performs GD on all training examples,
X: Training data set,
y: Labels for training data,
W: Weights vector,
B: Bias variable,
alpha: The learning rate,
max_iters: Maximum GD iterations.
'''
dW = 0 # Weights gradient accumulator
dB = 0 # Bias gradient accumulator
m = X.shape[0] # No. of training examples
for i in range(max_iters):
dW = 0 # Reseting the accumulators
dB = 0
for j in range(m):
# 1. Iterate over all examples,
# 2. Compute gradients of the weights and biases in w_grad and b_grad,
# 3. Update dW by adding w_grad and dB by adding b_grad,
W = W - alpha * (dW / m) # Update the weights
B = B - alpha * (dB / m) # Update the bias return W, B # Return the updated
weights and bias.
```