

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по научно-исследовательской работе
Тема: Разработка методики обнаружения аномалий в данных
космических лучей

Студент гр. 7303

Пэтайчук Н.Г.

Руководитель

Мандрикова Б.С.

Санкт-Петербург

2022

ЗАДАНИЕ НА НАУЧНО-ИССЛЕДОВАТЕЛЬСКУЮ РАБОТУ

Студент Пэтайчук Н.Г.

Группа 7303

Тема НИР: Разработка методики обнаружения аномалий в данных космических лучей

Задание на НИР:

Реализация и обучение нейронной сети для классификации данных нейтронного монитора на предмет наличия аномалий.

Сроки выполнения НИР: 01.09.2022 – 20.12.2022

Дата сдачи отчета: 18.12.2022

Дата защиты отчета: 27.12.2022

Студент

Пэтайчук Н.Г.

Руководитель

Мандрикова Б.С.

АННОТАЦИЯ

В рамках данного семестра выполнен один из этапов научно-исследовательской работы, а именно реализация и обучение нейронных сетей, предназначенных для классификации данных нейтронного монитора. В отчете описаны основные шаги по уменьшению размерности матриц и обоснование этого, реализации и обучении модифицированной сети LeNet, а также рассмотрение сети AlexNet для решения задачи классификации данных нейтронных монитора. В конце отчета представлено описание предполагаемого решения, которое будет реализовано в рамках весеннего семестра.

SUMMARY

During this semester, one of the stages of research work was completed, namely, the implementation and training of neural networks designed to classify neutron monitor data. The report describes the main steps to reduce the dimension of matrices and the rationale for this, the implementation and training of the modified LeNet network, as well as the consideration of the AlexNet network to solve the problem of classification of neutron monitor data. At the end of the report, a description of the proposed solution that will be implemented during the spring semester is presented.

СОДЕРЖАНИЕ

	Постановка задачи	5
1.	Результаты работы в осеннем семестре	8
1.1.	Уменьшение размера матриц	9
1.2.	Реализация и обучение модифицированной сети LeNet для классификации данных нейтронных мониторов	12
1.3.	Рассмотрение архитектуры AlexNet для классификации данных нейтронных мониторов	15
1.4.	Описание предполагаемого метода решения	18
2.	План работы на весенний семестр	21
	Заключение	22
	Список использованных источников	23

ПОСТАНОВКА ЗАДАЧИ

Одним из основных объектов исследования астрофизики являются космические лучи (КЛ). Основными источниками космических лучей являются звёзды и происходящие в них реакции, однако основным объектом исследования природы космических лучей является Солнце как ближайшая к нам звезда и источник термоядерных реакций [2]. Различают как первичные космические лучи, то есть частицы, находящиеся в космическом пространстве, так и вторичные, являющиеся продуктом взаимодействия первичных КЛ с атмосферой Земли (в результате таких реакций порождаются широкие атмосферные ливни, которые можно наблюдать на расстоянии тысяч метров) [3].

Поток частиц, приходящих на Землю из космического пространства, непостоянен и зависит как от времени, так и от места наблюдения за космическими лучами [2]. Изменения потока космических лучей во времени и в пространстве называют вариациями космических лучей, и основную роль в их формировании играет Солнце. Различают периодические вариации, к которым относят 22-летние, 11-летние, 27-дневные и солнечно-суточные вариации, которые связаны с солнечной активностью и которые можно наблюдать и регистрировать с заданной периодичностью [3]. С другой стороны, существуют и непериодические вариации к ним относят эффект Форбуша и солнечные вспышки. Хотя вероятность их наступления и повышается с ростом активности Солнца, но точно предсказать их пока не удаётся, что в свою очередь является проблемой, так как космические лучи влияют на работу как человеческого организма (в особенности организма космонавтов), так и электронику: системы теле-, радио и спутниковой связи, системы GPS и ГЛОНАСС, электрические сети, нефтегазовые комплексы и т.д. Непредвиденные резкие изменения интенсивности космических частиц могут представлять из себя угрозу для жизни и здоровья людей, а также для функционирования электронной техники [1, 2]. Существующие методы прогнозирования интенсивности космических лучей и

предотвращения негативных последствий не являются достаточно эффективными, что определяет *актуальность* работы.

Одним из основных наземных приборов наблюдения за космическими лучами являются нейтронные мониторы [4], чей принцип работы основывается на генерации нейтронов приходящими нуклонами и ядрами в результате ядерной реакции и дальнейшем подсчёте образовавшихся нейтронов. Примеры данных, получаемых с нейтронных мониторов, показаны на рис. 1. Как можно заметить, полученные данные можно интерпретировать как функцию интенсивности космических лучей от времени, благодаря чему её можно анализировать на предмет выявления непериодических вариаций. Разработка такого метода определения непериодических вариаций и является основной задачей данной работы.

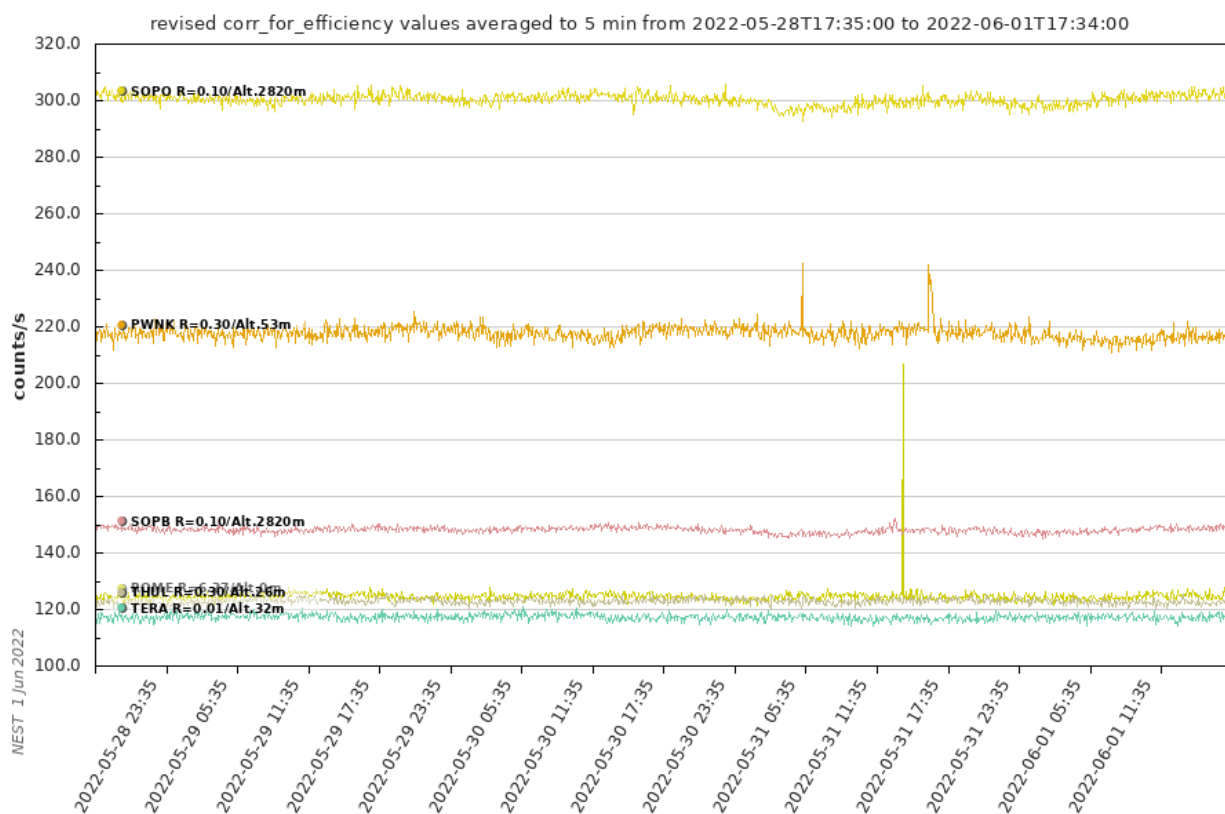


Рисунок 1 – Примеры данных, получаемых с нейтронных мониторов. По оси абсцисс подписаны временные метки, по оси ординат – число фиксируемых нейтронов в секунду.

Цвета обозначают нейтронные мониторы, с которых пришли данные

Целью данной работы является разработка методики обнаружения аномалий в сигнале космических лучей, по данным нейтронных мониторов.

Задачи данной работы:

1. Обзор существующих методов обработки данных с нейтронных мониторов;
2. Сравнительный анализ и выбор математических аппаратов, на основе которых будет разрабатываться метод идентификации неперiodических вариаций;
3. Выбор инструментов и технологий для реализации метода идентификации неперiodических вариаций;
4. Подготовка данных для настройки и проверки корректности работы метода идентификации неперiodических вариаций;
5. Реализация метода идентификации неперiodических вариаций;
6. Оценка эффективности полученной реализации метода идентификации неперiodических вариаций.

1. РЕЗУЛЬТАТЫ РАБОТЫ В ОСЕННЕМ СЕМЕСТРЕ

В качестве планов на осенний семестр в отчёте по научно-исследовательской работе были представлены следующие задачи:

1. Обучение нейронной сети по сформированным данным при описанной архитектуре;
2. Анализ полученных в ходе обучения свёрточной нейронной сети результатов. В случае, если она показывает себя неудовлетворительно в плане точности классификации, выбор другой архитектуры и/или формирование иных исходных данных;
3. Планирование архитектуры и реализация приложения на базе обученной нейронной сети для идентификации аномалий в данных космических лучей.

Однако в ходе выполнения задачи 1 возникли проблемы, требующие выполнения действий, которые описаны в разделе 1.1, что также привело к изменению архитектуры нейронной сети, описанной в предыдущем отчёте по научно-исследовательской работе. Тем не менее с помощью ещё некоторых дополнительных изменений архитектуры LeNet и применения метода кросс-валидации удалось добиться точности предсказания на тестовом наборе данных 90%. Помимо этого, была рассмотрена архитектура AlexNet на предмет того, можно ли при помощи неё добиться ещё большей точности предсказания. Также была разработана архитектура приложения, которое будет представлять из себя десктопное приложение, написанное на языке программирования Python и с применением библиотеки PyQt (подробнее об этом изложено в разделе 1.4).

Блокноты (файлы формата .ipynb) с реализацией архитектур свёрточных нейронных сетей модифицированной LeNet и AlexNet https://github.com/NikitaPetaichuk/petaichuk_research_work в папке `neural_networks`.

1.1. Уменьшение размера матриц

Изначально предполагалось работать с матрицами размером 2327x1438, однако для обучения нейронной сети с помощью матриц такого размера требуются как хорошие вычислительные мощности (GPU или TPU), так и достаточное количество оперативной памяти для хранения матриц и промежуточных результатов вычисления. Поэтому возникла мысль о том, чтобы уменьшить размерность матриц, по которым будет обучаться нейронная сеть.

В то же время встаёт вопрос о том, не потеряется ли точность представляемых данных, ведь тут идёт речь о природных данных. Можно точно сказать, что уменьшение числа столбцов приведёт к уменьшению точности представления данных, так как число столбцов зависит от временного интервала замера количества нейтронов (каждую минуту, каждые две минуты и так далее), а, как было показано в предыдущем отчёте по научно-исследовательской работе, вейвлет-образы, полученные с 5-минутных данных, серьёзно отличаются от вейвлет-образов, полученных с ежеминутных данных. Единственной возможностью уменьшить размерность матриц без потери точности представления является уменьшение числа строк, то есть уменьшения масштаба рассматриваемых частот, за что отвечает 2 параметр в переменной `scales` в Matlab-скриптах `do_wavelet.m` и `extract_data.m`, которые находятся в вышеуказанном Github-репозитории в папке `processing_extract_wavelet_data`.

Были проведены эксперименты с помощью скрипта Matlab-скрипта `do_wavelet.m` по проверке того, отличаются ли вейвлет-образы, для которых число строк уменьшили в 10 раз при помощи изменения 2 параметра переменной `scales` с 1 на 10, от изначальных вейвлет-образов. Как видно из рис. 2 и рис. 3, принципиальных отличий в вейвлет-образах не наблюдается, а это значит, что вместо матриц 2327x1438 можно использовать матрицы 233x1440 (число столбцов было увеличено до 1440 для представления данных за полный день или 1440 минут). На основании вышесказанного, с помощью скрипта `extract_data.m`

был сформирован новый массив данных, в которых вместо матриц 2327x1438 используются матрицы 233x1440.

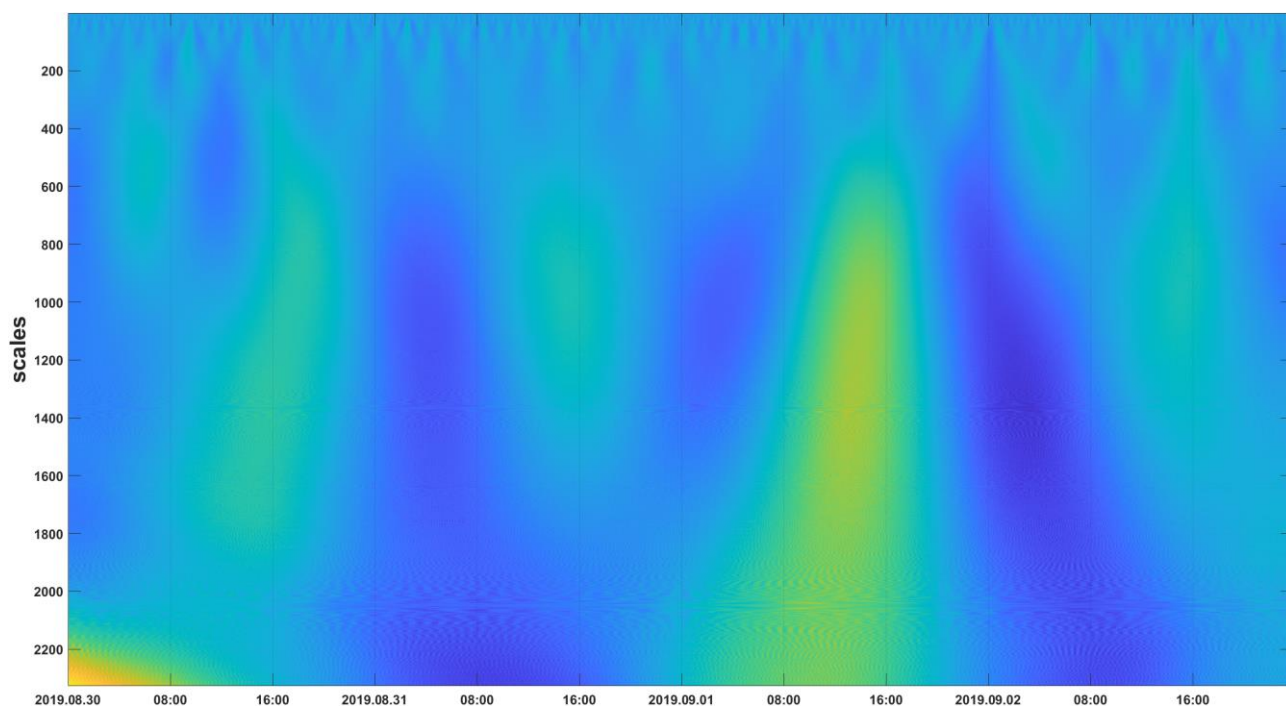


Рисунок 2 – Исходный вейвлет-образ, сформированный по данным за временной промежуток с 30.08.2019 по 02.09.2019 включительно

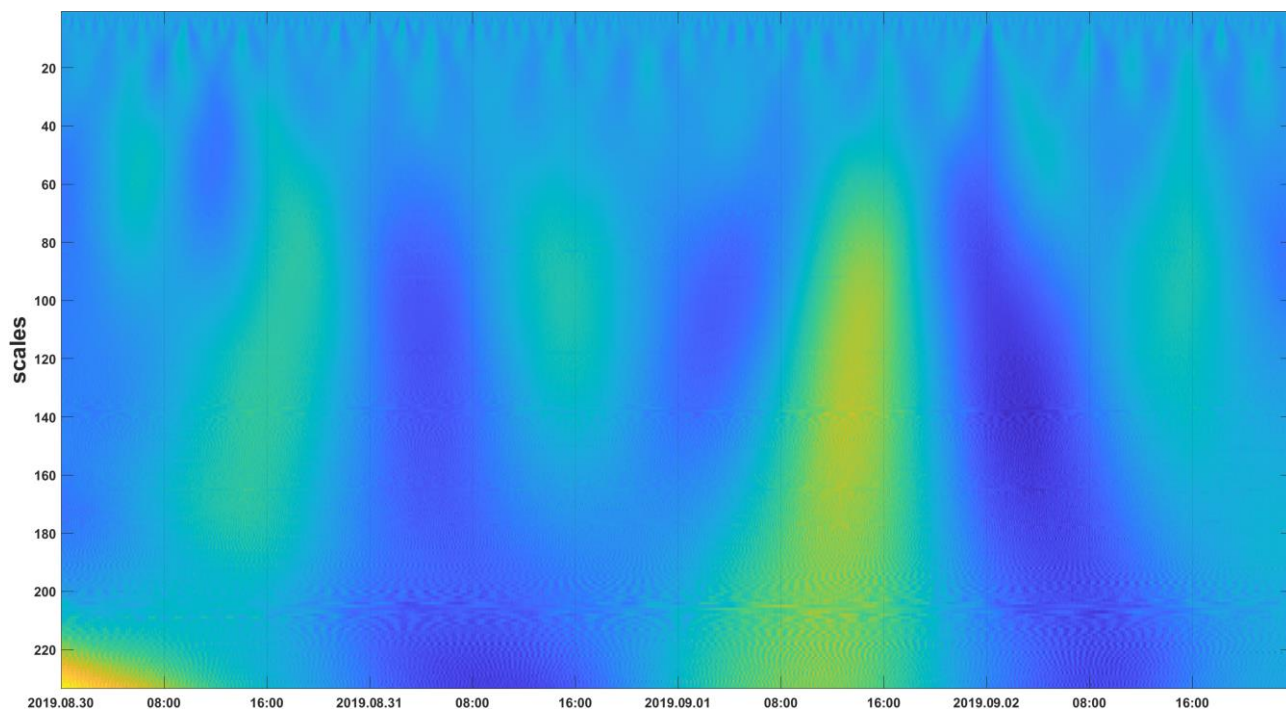


Рисунок 3 – Изменённый вейвлет-образ, сформированный по данным за временной промежуток с 30.08.2019 по 02.09.2019 включительно

Изменение размера матриц для обучения также требует корректировок для архитектуры нейронной сети, которая будет обучаться на новым данным. Были внесены следующие изменения в архитектуру:

- Входной слой – размер 233x1440,
- Ядро фильтра для свёртки – размер 36x36,
- После первой свёртки – 6 карт признаков размером каждая по 198x1405,
- После первой субдискретизации – 6 карт признаков размером каждая по 99x702,
- После второй свёртки – 16 карт признаков размером каждая по 64x667,
- После второй субдискретизации – 16 карт признаков размером каждая по 32x333,
- Два полносвязных слоя из 120 и 84 элементов (без изменений) и выходной слой из 3 элементов.

На рис. 4 представлена в графическом виде изменённая архитектура нейронной сети.

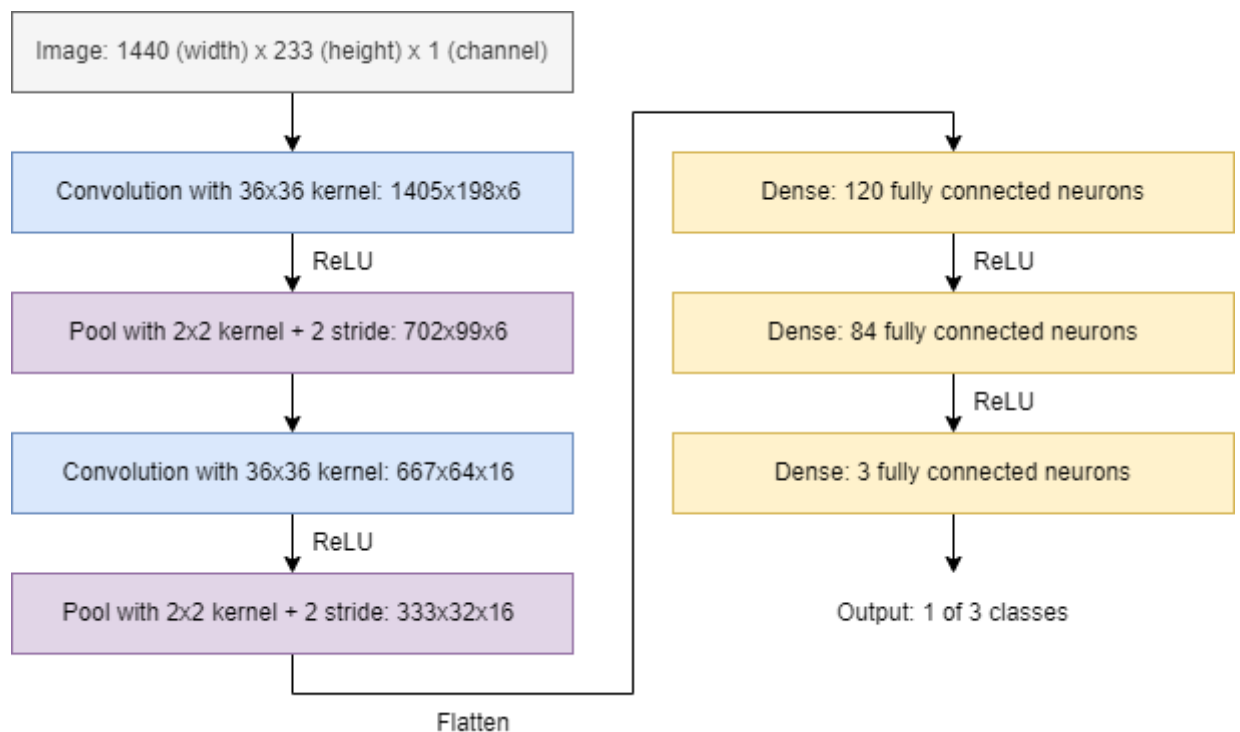


Рисунок 4 – Изменённая архитектура свёрточной нейронной сети

1.2. Реализация и обучение модифицированной сети LeNet для классификации данных нейтронных мониторов

Для реализации модифицированной архитектуры LeNet был выбран язык программирования Python и фреймворк PyTorch. Для обучения нейронной сети использовался веб-сервис Kaggle, где был создан датасет `sopo_neutron_monitor_data` на основе полученного ранее набора матриц, распределённых по папкам в зависимости от своего класса (спокойный день, слабая буря или сильная буря). Блокнот с модифицированной архитектурой LeNet представлен в Github-репозитории под именем `neutron_monitor_data_lenet.ipynb` в папке `neural_networks`.

Был реализован класс `NeutronMonitorDataset`, который реализует интерфейс стандартного класса `Dataset` фреймворка PyTorch по получению доступа к данным. В рамках класса `NeutronMonitorDataset` реализована «ленивая» загрузка данных, то есть при инициализации объекта класса формируется список путей к файлам с матрицами в датасете `sopo_neutron_monitor_data` и список соответствующих меток, а уже при попытке получения самой матрицы (вызов метода `__getitem__`) идёт загрузка матрицы по пути из списка, после чего возвращается преобразованная в тензор матрица (требуется для обучения) и метка этой матрицы. Исходный код класса `NeutronMonitorDataset` представлен на рис. 5.

Для итерации по данным в ходе обучения использовался стандартный класс `DataLoader`, в частности три объекта этого класса: для тренировочных, валидационных и тестовых данных. Объект класса `NeutronMonitorDataset` отвечает за доступ ко всем данным, поэтому для разделения данных на тренировочные, валидационные и тестовые использовалась функция `random_split`. Деление на вышеобозначенные категории шло в соотношении 70 : 20 : 10, и, как итог, размер тренировочных данных составляет 66 матриц, валидационных данных – 19, тестовых данных – 10 (всего матриц 95).

```

# Defining a class for neutron monitor dataset
class NeutronMonitorDataset(Dataset):
    def __init__(self):
        super().__init__()
        self.wavelets_files, self.labels = NeutronMonitorDataset._get_neutron_monitor_data_files()
        self.length = len(self.wavelets_files)

    @staticmethod
    def _get_neutron_monitor_data_files():
        wavelets_files, labels = [], []
        data_dirs_paths = [
            "../input/sopo-neutron-monitor-data/calm_days",
            "../input/sopo-neutron-monitor-data/weak_storms",
            "../input/sopo-neutron-monitor-data/strong_storms"
        ]
        for index, dir_path in enumerate(data_dirs_paths):
            wavelets_array, labels_array = NeutronMonitorDataset._get_data_files_from_directory(dir_path, index)
            wavelets_files.extend(wavelets_array)
            labels.extend(labels_array)
        return wavelets_files, labels

    @staticmethod
    def _get_data_files_from_directory(dir_path, label):
        wavelets_files_array = []
        for _, _, files in os.walk(dir_path):
            for file in files:
                if file != ".gitkeep":
                    file_path = os.path.join(dir_path, file)
                    wavelets_files_array.append(file_path)
        labels_array = [label for _ in range(len(wavelets_files_array))]
        return wavelets_files_array, labels_array

    def __len__(self):
        return self.length

    def __getitem__(self, index):
        sample_wavelet_file = self.wavelets_files[index]
        sample_wavelet_image = torch.tensor(
            np.loadtxt(sample_wavelet_file, dtype=np.float64, delimiter=','),
            dtype=torch.float32
        )
        sample_label = self.labels[index]
        return sample_wavelet_image, sample_label

```

Рисунок 5 – Исходный код класса NeutronMonitorDataset

Для реализации непосредственно модифицированной архитектуры был создан класс NeutronMonitorDataLeNet, содержащий два атрибута: один, feature_extractor, отвечает за все операции свёртки и пулинга (от англ. «pooling»), а второй, classifier, содержит все полносвязные слои нейтронной сети. Метод forward есть метод, который отвечает за применение нейронной сети к входным данным (так называемый «прямой ход»), то есть в данном случае он по полученной входной матрице вернёт вероятности принадлежности к конкретному классу. Исходный код класса NeutronMonitorDataLeNet представлен на рис. 6.

```
# defining modified LeNet model for neutron monitor data
class NeutronMonitorDataLeNet(nn.Module):
    def __init__(self):
        super(NeutronMonitorDataLeNet, self).__init__()
        self.feature_extractor = nn.Sequential(
            nn.Conv2d(in_channels=1, out_channels=6, kernel_size=36),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=3),
            nn.Conv2d(in_channels=6, out_channels=16, kernel_size=36),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=3),
            nn.Flatten(0)
        )
        self.classifier = nn.Sequential(
            nn.Linear(in_features=23040, out_features=120),
            nn.ReLU(),
            nn.Linear(in_features=120, out_features=84),
            nn.ReLU(),
            nn.Linear(in_features=84, out_features=3)
        )

    def forward(self, x):
        x = self.feature_extractor(x)
        # print(x.shape)
        logits = self.classifier(x)
        probs = F.softmax(logits, dim=0)
        return logits, probs
```

Рисунок 6 – Исходный код класса NeutronMonitorDataLeNet

Для обучения нейронной сети была создана функция `training_loop`, которая сначала проводит цикл обучения с помощью созданной функции `train`, потом проверяет степень обучения нейронной сети с помощью функции `validate`, после чего выводит информацию по эпохе (значение функции потерь и точность предсказания). Также была создана функция `training_kfold_loop`, которая благодаря классу `KFold` из библиотеки `sklearn` реализует обучение нейронной сети с применением метода кросс-валидации. Обучение происходило на GPU P100 16 GB, в качестве функции потерь использовалась кросс-энтропия, в качестве оптимизатора использовался метод адаптивной оценки моментов (или Adam), число эпох равно 15. Для минимизации случайности в процессе обучения был установлен общий постоянный сид для случайной генерации с помощью вызова функции PyTorch `manual_seed` и для разделения всего набора данных на тренировочные, валидационные и тестовые с помощью задания значения аргумента `generator` функции `random_split` объектом класса `torch.Generator` с заданным постоянным сидом.

К сожалению, модифицированная архитектура LeNet не смогла показать удовлетворительных результатов по точности предсказания (70% и более). В

случае обучения без применения кросс-валидации точность на тренировочных данных после 15 эпох достигла 43,94%, а точность на валидационных данных – 26,32%. В свою очередь при использовании кросс-валидации точность на тренировочных и валидационных данных в конце обучения оказалась равной 36,76% и 29,41% соответственно, при этом среднее значение точности за всё время обучения составило 39,96% и 37,96% соответственно для тренировочных и валидационных данных. В обоих случаях обучения точность на тестовых данных составила 50%. Для повышения точности были предприняты различные изменения параметров архитектуры, в частности размера ядра свёртки, и проверкой того, насколько изменилась точность при новой архитектуре.

Наилучшие результаты обучения были получены при уменьшении размера ядер свёртки до 3. В этом случае без кросс-валидации точность предсказаний на тренировочных и валидационных данных составила под конец обучения 100% и 42,11% соответственно, а точность на тестовых данных составила 70%, что является удовлетворительным результатом. Однако при использовании кросс-валидации точность и на тренировочных, и на валидационных данных под конец обучения составляла 100%, а на тестовых данных точность составила 90%. Таким образом можно утверждать, что обучение нейронной сети прошло успешно, однако возникла мысль рассмотреть более продвинутую архитектуру, а именно AlexNet, на предмет того, улучшит ли ещё её использование точность предсказаний.

1.3. Рассмотрение архитектуры AlexNet для классификации данных нейронных мониторов

Архитектура AlexNet обладает большим числом операций свёрток и пулинга, как показано на рис. 7, а значит и способность извлекать признаки у такой архитектуры больше, нежели чем у LeNet. При этом саму архитектуру не требуется модифицировать, так как используемый изначально размер изображения 224x224 близок к числу строк у используемых для обучения

матриц. Единственным отличием будет то, что в последнем полносвязном слое будет 3 нейрона вместо 1000.

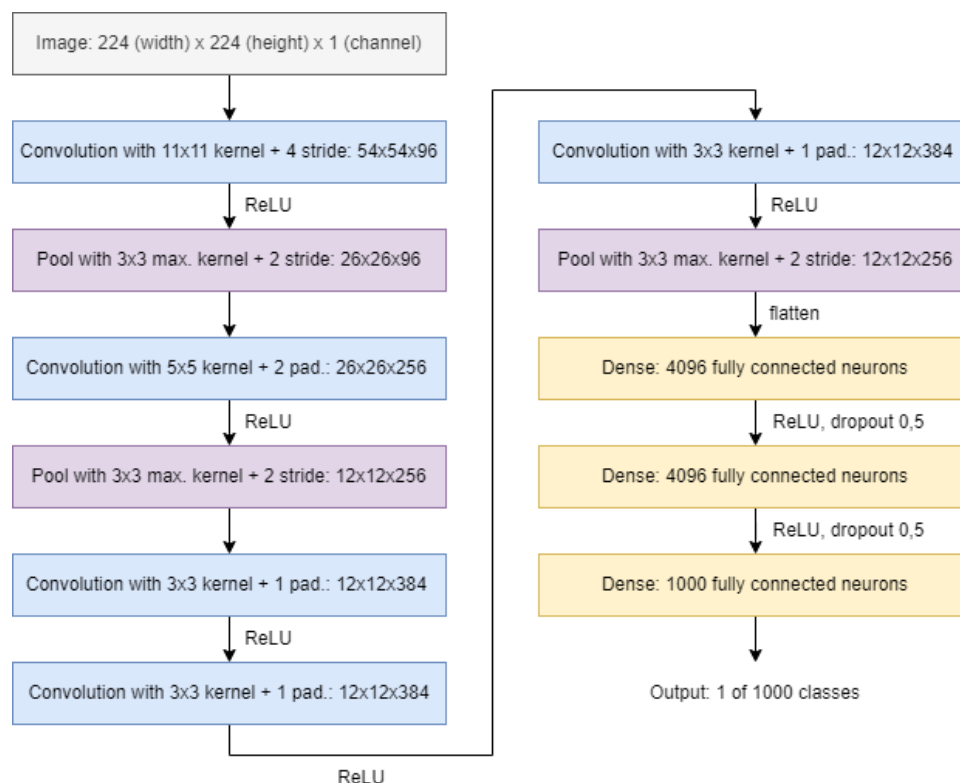


Рисунок 7 – Архитектура AlexNet

Для внедрения архитектуры AlexNet в существующий код необходимо лишь изменить класс `NeutronMonitorDataLeNet`, а точнее его атрибуты `feature_extractor` и `classifier` в соответствии с архитектурой AlexNet. Исходный код изменённого класса `NeutronMonitorDataLeNet` показан на рис. 8, весь код с предпринятыми изменениями был вынесен в отдельный блокнот `neutron_monitor_data_alexnet.ipynb`, который находится в папке `neural_networks` в Github-репозитории. Обучалась нейронная сеть на базе архитектуры AlexNet в двух вариантах: с использованием исходной архитектуры AlexNet и с использованием архитектуры AlexNet, в котором размер всех ядер свёрток равен 3. При этом для этих двух случаев было проведено обучение с использованием кросс-валидации и без него.


```

# defining modified LeNet model for neutron monitor data
class NeutronMonitorDataLeNet(nn.Module):
    def __init__(self):
        super(NeutronMonitorDataLeNet, self).__init__()
        self.feature_extractor = nn.Sequential(
            nn.Conv2d(in_channels=1, out_channels=64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(in_channels=64, out_channels=192, kernel_size=5, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(in_channels=192, out_channels=384, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(in_channels=384, out_channels=256, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.Conv2d(in_channels=256, out_channels=256, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Flatten(0)
        )
        self.classifier = nn.Sequential(
            nn.Dropout(p=0.5, inplace=False),
            nn.Linear(in_features=67584, out_features=4096),
            nn.ReLU(),
            nn.Dropout(p=0.5, inplace=False),
            nn.Linear(in_features=4096, out_features=4096),
            nn.ReLU(),
            nn.Linear(in_features=4096, out_features=3)
        )

    def forward(self, x):
        x = self.feature_extractor(x)
        logits = self.classifier(x)
        probs = F.softmax(logits, dim=0)
        return logits, probs

```

Рисунок 8 – Исходный код изменённого класса NeutronMonitorDataLeNet

Результаты обучения двух версий нейронной сети AlexNet с использованием кросс-валидации и без неё, а именно точность предсказания на тестовых данных, представлены на табл. 1, где также представлены данные по обучению нейронной сети LeNet для сравнения. Из таблицы видно, что использование AlexNet для решения задачи классификации данных, полученных с нейтронного монитора, менее целесообразно по сравнению с LeNet с ядром свёртки 3x3 и что AlexNet в любой из двух конфигураций находится на одном уровне с изначальной версией LeNet, где размер ядра свёртки составляет 36x36. Таким образом, было получено подтверждение тому, что для решения задачи классификации данных, полученных с нейтронного монитора, стоит использовать ту версию архитектуры LeNet, в которой используются ядра свёртки 3x3, а значит можно приступить к реализации решения, базирующегося на этой нейронной сети. С другой стороны, целесообразно также провести дополнительное тестирование на большем количестве данных, а то полученная точность предсказаний в 90% была получена только на 10 матрицах и потому может являться грубой оценкой точности.

Нейронная сеть	Точность без кросс-валидации	Точность с кросс-валидацией
LeNet (ядро свёртки 36x36)	50%	50%
LeNet (ядро свёртки 3x3)	70%	90%
AlexNet (оригинальная)	50%	50%
AlexNet (ядро свёртки 3x3)	50%	50%

Таблица 1 – Полученные точности предсказаний для разных вариантов нейронных сетей

1.4. Описание предполагаемого метода решения

Предполагаемое решение будет в формате десктопного приложения. Поскольку главной его составляющей будет нейронная сеть, реализованная на языке программирования Python с применением фреймворка PyTorch, то и реализовывать всё приложение имеет смысл с использованием библиотеки PyQt, что в свою очередь обеспечит его кроссплатформенность. Приложение при запуске будет подгружать хранящиеся в отдельном файле параметры нейронной сети, полученные после её успешного обучения. Помимо этого, на язык программирования Python в целях внедрения в приложение будет переписаны операции по вейвлет-преобразованию, которые на текущий момент представлены в формате Matlab-скриптов. Предполагается, что для этих целей будет использоваться библиотека PyWavelets. Также на языке Python будут реализованы дополнительные операции по предобработке данных, если они потребуются.

Макет предполагаемого решения представлен на рис. 9. Типичный сценарий использования приложения заключается в следующем:

1. Выбор файла для анализа:
 - а. Пользователь вводит в поле ввода путь до файла в формате `<временная_метка>;<число_зафиксированных_нейтронов>`, в

котором хранятся данные, полученные с нейтронного монитора за конкретный день плюс за 2 дня до и за 1 день после;

в. Либо пользователь может нажать кнопку «Обзор», чтобы в появившемся окне выбрать нужный файл;

2. Пользователь нажимает на кнопку «Обнаружить аномалию», после чего ожидает обработки данных из файла, которая заканчивается выводом результата о наличии бурь в дань, данные по которому были проанализированы, плюс выводом вейвлет-образа, сформированного по полученным данным;
3. Пользователь может при желании сохранить полученный вейвлет-образ путём нажатия на кнопку «Сохранить вейвлет-образ» и последующим вводом имени файла и выбором местоположения в открывшемся окне.

Обработка данных при нажатии кнопки «Обнаружить аномалию» будет заключаться в выполнении следующих шагов:

1. Чтение данных из файла, проверка их на корректность формата;
2. Выполнение вейвлет-преобразования считанных данных, получение вейвлет-образа в форме матрицы;
3. «Вырезание» необходимой части вейвлет-образа, в котором представлены данные за рассматриваемый день;
4. Применение полученной матрицы в качестве входных данных для нейронной сети;
5. На основе наибольшей из полученных вероятностей принадлежности к конкретному классу вывести на экран сообщение о наличии бурь («Спокойный день, никаких космических бурь», «Неспокойный день, слабые бури» или «Неспокойный день, сильные бури»), а также вывести на экран вейвлет-образ.

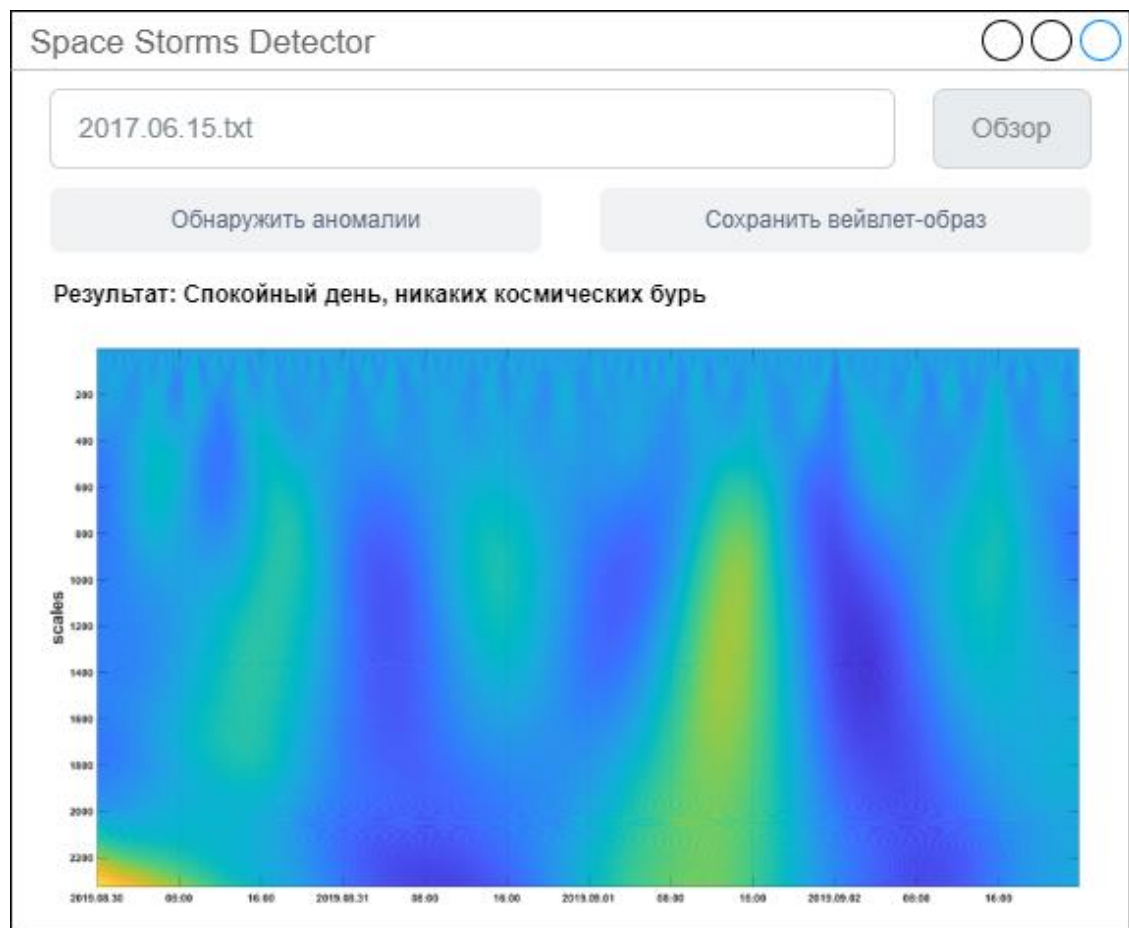


Рисунок 9 – Макет предполагаемого решения

2. ПЛАН РАБОТЫ НА ВЕСЕННИЙ СЕМЕСТР

По результатам выполнения задач за осенний семестр был построен план работы на весенний семестр, который включает в себя следующие задачи:

1. Протестировать на большем наборе тестовых данных нейронную сеть с архитектурой LeNet, использующей ядра свёртки размером 3x3;
2. На основе макета и параметров обученной нейронной сети реализовать решение в виде десктопного приложения;
3. Проанализировать параметры работы полученного решения на предмет скорости работы и других характеристик.

ЗАКЛЮЧЕНИЕ

В ходе работы за осенний семестр были рассмотрены модифицированная архитектура LeNet и архитектура Alex в целях их использования для решения задачи классификации данных, полученных с нейтронного монитора. Была рассмотрена возможность уменьшения размера матриц для их использования в целях обучения нейронной сети, а также были предприняты попытки обучения нейронных сетей с вышеобозначенными архитектурами с использованием кросс-валидации. Было получено, что наилучшую точность предсказаний в рамках рассматриваемой задачи смогла показать нейронная сеть LeNet с ядрами свёртки 3x3 и что она превосходит в этом плане даже архитектуру AlexNet. Помимо прочего, было представлено описание предполагаемого, базирующегося на нейронной сети, предназначенной для классификации данных, полученных с нейтронного монитора.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Мирошниченко Л. И. Физика Солнца и солнечно-земных связей: учебное пособие // Л. И. Мирошниченко; Под ред. М. И. Панасюка. — М.: Университетская книга, 2011, 345 с.
2. Мурзин В.С. Астрофизика космических лучей: Учебное пособие для вузов. — М.: Университетская книга; Логос, 2007, 488 с.
3. Топтыгин И.Н. Космические лучи в межпланетных магнитных полях. — М.: Наука, 1983, 304 с.
4. Real time data base for the measurements of high-resolution Neutron Monitor. [Электронный ресурс]. www.nmdb.eu (дата обращения 15.03.2022).