

Федеральное государственное бюджетное  
образовательное учреждение высшего образования



«Московский государственный технический  
университет имени Н. Э. Баумана (национальный  
исследовательский университет)»



Факультет «Машиностроительные технологии»

Кафедра «Электронные технологии в  
машиностроении»

## ОТЧЁТ

по дисциплине «Анализ и синтез технических решений»

на тему:

«Анализ характеристик фотонного кристалла с титановым покрытием  
методами машинного обучения»

Выполнил:

студент группы МТ11-72Б  
Зотов М.С.

Руководитель:

Колесник Л.Л.

Дисциплина	Результат защиты (нужно выделить)	
Научно-исследовательская работа	НЕЗАЧЁТ	ЗАЧЁТ
Зачёт принял _____ подпись	(_____)	расшифровка

Москва

2021 г.

# **РЕФЕРАТ**

Отчёт по дисциплине «Анализ и синтез технических решений» на тему:  
**«Анализ характеристик фотонного кристалла с титановым покрытием  
методами машинного обучения»**

Расчёто-пояснительная записка содержит 40 страниц, 12 таблиц, 12 рисунков, 34 уравнения, список литературы из 10 источников.

**ФОТОННЫЙ КРИСТАЛЛ, ПОЛНЫЙ ФАКТОРНЫЙ ЭКСПЕРИМЕНТ,  
ТРЁХМЕРНЫЙ ФОТОННЫЙ КРИСТАЛЛ, НЕЙРОННАЯ СЕТЬ, АЛГОРИТМ  
ОПТИМИЗАЦИИ, ОБРАТНОЕ РАСПРОСТРАНЕНИЕ ОШИБКИ, СКРЫТЫЕ  
СЛОИ, ADAM, РАЗМЕРНОСТЬ ДАННЫХ**

Работа посвящена анализу методов нейросетевой обработки результатов.

Построена математическая модель зависимости ширины запрещённой зоны от мощности, подаваемой на магнетрон, и времени нанесения.

Разработана нейросетевая структура для предсказания результатов последующих экспериментов точностью 92%.

Проанализировано влияние размерностей входных и выходных данных на качество модели.

Исследована зависимость точности сети от количества нейронов в скрытом слое.

Изучено влияние функции активации на результаты модели.

Проверена зависимость параметров модели от алгоритма оптимизации.

Математическое моделирование выполнено инструментами GNU Octave и LibreOffice Calc. Нейросетевое моделирование проведено при помощи Jupyter Notebook, IPython, scikit-learn и NumPy. Визуализация выполнена при помощи matplotlib и gnuplot. Записка выполнена с использованием среды L<sup>A</sup>T<sub>E</sub>X.

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ . . . . .	4
1 Математическое моделирование процесса нанесения проводящего слоя методом магнетронного распыления . . . . .	5
1.1 Анализ факторов технологической операции . . . . .	5
1.2 Выбор наиболее существенного выходного параметра . . . . .	6
1.3 Выбор наиболее существенных входных факторов . . . . .	7
1.4 Проведение математического моделирования технологического процесса . . . . .	7
1.4.1 Обоснование необходимости проведения математического моделирования технологического процесса . . . . .	7
1.4.2 Разработка плана эксперимента . . . . .	8
1.4.3 Построение математической модели . . . . .	9
1.4.4 Обработка результатов эксперимента . . . . .	10
1.4.4.1 Выборочное среднее и дисперсия . . . . .	10
1.4.4.2 Проверка эксперимента на воспроизводимость . . . . .	10
1.4.4.3 Определение коэффициентов полинома . . . . .	11
1.4.4.4 Оценка значимости коэффициентов . . . . .	11
1.4.4.5 Проверка модели на адекватность . . . . .	12
1.5 Моделирование интенсивности и длины отражённой волны . . . . .	14
1.5.1 Длина отражённой волны . . . . .	14
1.5.2 Интенсивность отражения . . . . .	15
1.5.3 Результаты по 3 экспериментам . . . . .	16
1.6 Анализ результатов эксперимента и выводы . . . . .	16
2 Машинное обучение в Jupyter Notebook . . . . .	19
2.1 Добавление данных . . . . .	19
2.2 Нормализация данных . . . . .	20
2.3 Выбор оптимальных параметров для обучения . . . . .	20

2.3.1	Методы контроля . . . . .	20
2.4	Влияние размерности сети на качество модели . . . . .	22
2.4.1	Транспортирование массивов . . . . .	23
2.4.2	Создание архитектуры модели . . . . .	23
2.4.3	Выводы . . . . .	26
2.5	Влияние числа скрытых слоёв сети на качество модели . . . . .	27
2.5.1	Выводы . . . . .	31
2.6	Влияние функции активации на результаты модели . . . . .	31
2.6.1	Выводы . . . . .	34
2.7	Влияние солвера на результаты модели . . . . .	34
2.7.1	Вывод . . . . .	35
2.8	Проверка полученной модели . . . . .	35
2.9	Получение результатов из модели . . . . .	36
ЗАКЛЮЧЕНИЕ	. . . . .	38
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	. . . . .	39

# **ВВЕДЕНИЕ**

**Нейросети** – перспективное направление современных исследований. С их помощью можно строить сложные модели исследуемых систем.

Моделирование процессов при помощи технологий машинного обучения позволяет значительно ускорить процесс обработки результатов исследований и оптимизировать процесс производства фотонных кристаллов.

В этой работе представлены методы создания нейросетей при небольшом количестве экспериментальных данных, подбор оптимальных размерностей входных и выходных данных, а также влияние функций активации и оптимизатора на выходные параметры.

**Цель данной работы** – анализ современных методов обработки экспериментальных данных при помощи нейронных сетей.

## **Задачи:**

- провести обработку результатов по методам техники эксперимента;
- создать нейронную сеть с помощью инструментов современного ПО;
- проанализировать влияние размерностей входных и выходных данных на качество модели;
- исследовать зависимость точности сети от количества нейронов в скрытом слое;
- изучить влияние функции активации на результаты модели;
- проверить зависимость параметров модели от алгоритма оптимизации.

# 1 Математическое моделирование процесса нанесения проводящего слоя методом магнетронного распыления

В предыдущей части были рассмотрены варианты формирования фотонного кристалла. Был выбран метод центрифугирования как наиболее быстрый. Были найдены наиболее эффективные параметры, при которых проходит формирования многослойной структуры. После получения массива сфер, их необходимо покрыть слоем титана. Слой будет выглядеть следующим образом (см. рис. 1) [1].

Расчёт, аналогичный данному, проводился авторами [2]

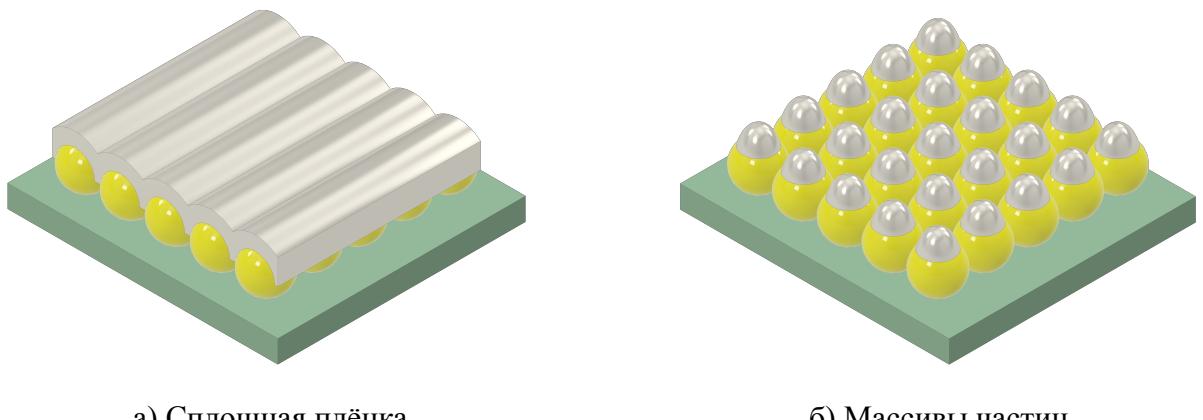


Рис. 1. Варианты структур, получаемых при осаждении тонкой пленки на оплавовую матрицу

## 1.1 Анализ факторов технологической операции

1. Входные контролируемые и управляемые факторы:
  - 1.1. Время напыления (варьируемый);
  - 1.2. Время очистки подложек в УЗВ (стабилизируемый);
  - 1.3. Мощность магнетрона (варьируемый);
  - 1.4. Режимы формирования коллоидной пленки (стабилизируемый);
  - 1.5. Материал подложки (стабилизируемый);
  - 1.6. Рабочее давление в камере ( $1 \cdot 10^{-1}$  мбар) (стабилизируемый);

1.7. Остаточное давление в камере ( $1 \cdot 10^{-3}$  мбар) (стабилизируемый);

1.8. Расход рабочего газа (стабилизируемый);

1.9. Расстояние от мишени до подложки (стабилизируемый).

**2. Входные контролируемые, но неуправляемые факторы:**

2.1. Напряжение на блоке питания;

2.2. Сила тока на блоке питания;

2.3. Температура магнетрона;

2.4. Температура подложки;

2.5. Время тренировки мишени.

**3. Входные неконтролируемые и неуправляемые факторы:**

3.1. Равномерность нанесения плёнки;

3.2. Количество аргона в потоке напускаемого газа;

3.3. Температура магнетрона;

3.4. Человеческий фактор;

3.5. Дефекты подложки;

3.6. Примеси со стенок камеры.

## **1.2 Выбор наиболее существенного выходного параметра**

В качестве выходного параметра (функции отклика) используются три величины:

1. Ширина Ф33;

2. Длина отраженной волны;

3. Интенсивность отражения.

т.к. комбинация этих трёх параметров даст наиболее полное представление о качестве полученной структуры.

## **1.3 Выбор наиболее существенных входных факторов**

Длина отражаемой световой волны (а также её интенсивность и ширина ФЗЗ) зависят не только от толщины полученной пленки, но и от множества других факторов, по которым теоретически сложно его определить, например, от режимов внедрения материала в пустоты (см. рис. 1).

Поэтому необходимо контролировать такие факторы, по которым нельзя или сложно произвести теоретические расчеты и которые с большой вероятностью будут влиять на рассматриваемый параметр. Поэтому, в качестве наиболее существенных входных факторов, были выбраны:

1. Время напыления;
2. Мощность магнетрона.

## **1.4 Проведение математического моделирования технологического процесса**

### **1.4.1 Обоснование необходимости проведения математического моделирования технологического процесса**

**Основная цель проведения эксперимента** – разработка математической модели, адекватно описывающей зависимость параметров фотонного кристалла (ширина ФЗЗ, длина отраженной волны, интенсивность отражения) от варьируемых факторов. Математическое описание процесса, обычно, представляется в виде полинома:

$$Y = b_0 + \sum_{j=1}^k b_j X_j + \sum_{j \neq u}^k b_{ju} X_j X_i + \sum_{j=1}^k b_j X_j^2 + \dots, \text{ где} \quad (1)$$

- $Y$  – функция отклика;
- $X_j$  – факторы исследуемого процесса.

## 1.4.2 Разработка плана эксперимента

1. В качестве метода исследования процесса выбран полный факторный эксперимент (ПФЭ). В этом случае учитывается влияние на функцию отклика исследуемого процесса не только каждого рассматриваемого в эксперименте фактора в отдельности, но и их взаимодействий [3].

2. Предположим, что модель имеет модель полинома первого порядка.

$$Y = b_0 + \sum_{j=1}^k b_j X_j + \sum_{j \neq u}^k b_{ju} X_j X_i \quad (2)$$

3. Число опытов найдём по формуле:

$$N = U^k = 2^2 = 4 \text{ (для линейной модели), где} \quad (3)$$

—  $u$  — число, на единицу большее порядка полинома

—  $k$  — число исследуемых факторов

4. Для линейной модели и 2 факторов достаточно будет провести 4 опыта. План эксперимента выглядит так, как показано на рис. 2. Модель примет вид:

$$Y = b_0 + b_1 X_1 + b_2 X_2 + b_{12} X_{12}, \text{ где} \quad (4)$$

—  $b_0$  — значение функции отклика в центре плана;

—  $b_1, b_2$  — коэффициенты, характеризующие степени влияния соответствующих факторов на функцию отклика;

—  $b_{12}$  — коэффициент, характеризующий влияние взаимодействия факторов.

5. Для построение математической модели, составим табл. 1

Таблица 1. Диапазоны варьирования входных факторов

Уровень	Мощность магнетрона, Вт	Время распыления, мин	В безразмерной системе координат
Верхний	200	10	+1
Нижний	150	5	-1

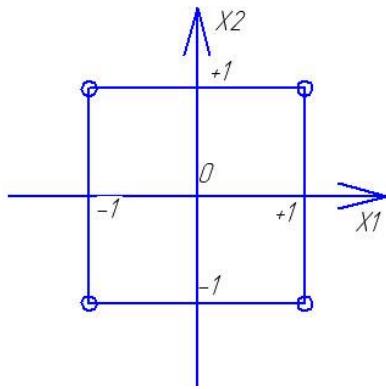


Рис. 2. Геометрическое изображение экспериментальных точек ПФЭ

6. В качестве центра плана принимается центр исследуемой области.

Значения входных параметров в центре плана:

- Мощность магнетрона  $W = 175$  Вт;
- Время распыления  $t = 7.5$  мин.

7. Выходной параметр измеряется в:

- ширина запрещённой зоны – нм;
- длина отражённой волны – нм;
- интенсивность отражения – %

### 1.4.3 Построение математической модели

Для рассматриваемого случая матрица планирования и проведения эксперимента имеет вид, представленный в табл. 2:

Таблица 2. Матрица планирования эксперимента по определению зависимости ширины запрещённой зоны от времени и мощности

№ опыта	Матрица планирования эксперимента										
	P	t	k	1	2	n	Y – ШЗЗ. нм		$S_i^2$	$Y_{\text{мод}}$	
							1	2			
1	150	5	1	-1	-1	1	50	57	53.5	24.5	53.5
2	150	10	1	-1	1	-1	51	50	50.5	0.5	50.5
3	200	5	1	1	-1	-1	58	55	56.5	4.5	56.5
4	200	10	1	1	1	1	36	40	38	8	38

## 1.4.4 Обработка результатов эксперимента

### 1.4.4.1 Выборочное среднее и дисперсия

Поскольку в каждом опыте было проведено 2 параллельных наблюдения определим:

1. Выборочное среднее:

$$\bar{Y}_i = \frac{\sum_{k=1}^n Y_{ik}}{n} \quad (5)$$

2. Выборочная дисперсия (число степеней свободы определяется число параллельных наблюдений в каждом опыте):

$$S_i^2 = \sum_{l=1}^n \frac{(Y_{ik} - \bar{Y}_i)^2}{n-1} \quad (6)$$

### 1.4.4.2 Проверка эксперимента на воспроизводимость

Проверку на воспроизводимость определим по критерию Кохрена:

1. Экспериментальное значение критерия Кохрена:

$$G_3 = \frac{\max(S_i^2)}{\sum_{i=1}^N S_i^2} = \frac{24.5}{24.5 + 0.5 + 4.5 + 8} = 0.65 \quad (7)$$

2. Критическое значение критерия Кохрена:

$$G_{kp} = G(\beta = 0.05, N = 4, n = 2) = 0.91, \text{ где} \quad (8)$$

- $\beta$  — уровень значимости;
- $N$  — число проведённых опытов;
- $n$  — число параллельных наблюдений.

3. Сравнение экспериментального и критического:

$$G_3 < G_{kp} \quad (9)$$

Таким образом дисперсии являются однородными, а эксперимент – воспроизводимым.

#### 1.4.4.3 Определение коэффициентов полинома

Коэффициенты полинома вычисляются по формуле:

$$b_j = \frac{\sum_{i=1}^N X_{ji} \bar{Y}_i}{N} \quad (10)$$

Значения приведены в 3.

Таблица 3. Значения коэффициентов полинома

$b_0$	$b_1$	$b_2$	$b_{12}$
49.625	-2.375	-5.375	-3.875

#### 1.4.4.4 Оценка значимости коэффициентов

Незначимость коэффициента может быть вызвана следующими причинами:

- интервал варьирования соответствующей переменной мал;
- уровень базового режима по данной переменной близок к точке частного экстремума;
- данный фактор не влияет на функцию отклика.

Проведём оценку значимости коэффициентов:

1. **Дисперсия воспроизводимости** (среднее арифметическое группы выборочных дисперсий (т.е. дисперсий функции отклика по каждому опыту)):

$$S^2(Y) = \frac{\sum_{i=1}^N S_i^2}{N} = \frac{24.5 + 0.5 + 4.5 + 8}{4} = 9.375 \quad (11)$$

**2. Дисперсия ошибки определения коэффициента:**

$$S^2(b_j) = \frac{S^2(Y)}{nN} = \frac{9.375}{2 \cdot 4} = 1.17 \quad (12)$$

**3. Оценка значимости по критерию Стьюдента**, значение которого рассчитывается по формуле:

$$t_j = \frac{|b_j - 0|}{\sqrt{S^2(b_j)}} \quad (13)$$

**4. Критическое значение критерия Стьюдента:**

$$t_{kp} = t(\beta = 0.1, \nu = N(n-1)) = t(0.1, 4) = 2.13 \quad (14)$$

**5. Отбрасывание незначимых коэффициентов:**

Таблица 4. Критерий Стьюдента. Оценка значимости факторов

i	$b_i$	$t_i$	$t_{kp}$	Значимость
0	49.625	45.84		Значим
1	-2.375	2.19		Значим
2	-5.375	4.96		Значим
12	-3.875	3.58		Значим

Получается, что в модели отсутствуют незначимые коэффициенты.

**6. Окончательный вид модели.** Из (4) и табл. табл. 4:

$$Y = 49.625 - 2.375X_1 - 5.375X_2 - 3.875X_1X_2 \quad (15)$$

#### 1.4.4.5 Проверка модели на адекватность

1. Для оценки адекватности модели необходимо определить **значения функции отклика** в каждом опыте согласно математической модели (15).

Таблица 5. Значения функции отклика, рассчитанные по математической модели

$\hat{Y}_1$	$\hat{Y}_2$	$\hat{Y}_3$	$\hat{Y}_4$
53.5	50.5	56.5	38

**2. Дисперсия адекватности** (оценка отклонения, предсказанного моделью значения выходного параметра (функции отклика) от результатов эксперимента в каждой точке факторного пространства):

$$S_{\text{ад}}^2 = \frac{\sum_{i=1}^N (\bar{Y}_i - \hat{Y}_i)^2}{N - \alpha_{3H}} n, \text{ где} \quad (16)$$

- $\alpha_{3H} = 4$  — число значимых коэффициентов в полиноме;
- $n$  — число параллельных измерений повторений.

Поскольку

$$N = \alpha_{3H} = 4 \quad (17)$$

необходимо провести дополнительный опыт.

Таблица 6. Дополнительный опыт

№	$X_0$	$X_1$	$X_2$	$X_1X_2$	$Y_1$	$Y_2$	$Y_{\text{ср}}$	$S_i^2$	$Y_{\text{мод}}$
5	1	0.5	0.5	0.25	40	44.6	42.3	10.58	44.78

Тогда формула (16) примет вид

$$S_{\text{ад}}^2 = \frac{(44.78 - 42.3)^2}{1} = 12.31 \quad (18)$$

**3. Проверка на адекватность при помощи критерия Фишера.** Возьмём отношение дисперсии адекватности (16) к дисперсии воспроизводимости (11):

$$F_3 = \frac{S_{\text{ад}}^2}{S^2(Y)} = \frac{12.31}{9.375} = 1.31 \quad (19)$$

**4. Критическое значение критерия Фишера:**

$$F_{\text{кр}} = F(\beta, N - \alpha_{3H}, N(n - 1)) = F(0.05, 1, 5) = 6.61 \quad (20)$$

**5. Сравнение экспериментального и критического:** из (19) и (20)

$$F_3 < F_{\text{кр}} \Rightarrow \text{модель адекватна} \quad (21)$$

Таким образом, полученная модель имеет вид, показанный в (15)

## 1.5 Моделирование интенсивности и длины отражённой волны

Аналогично моделированию ширины запрещённой зоны проведём моделирование интенсивности и длины отражённой волны (кратко в таблицах)

### 1.5.1 Длина отражённой волны

Таблица 7. Матрица планирования эксперимента по определению зависимости длины отражённой волны от времени и мощности

№	Матрица планирования эксперимента										
	P	t	k	1		2		n	Y – λ, нм		
				1	2	1	2		Y <sub>1</sub>	Y <sub>2</sub>	Y <sub>ср</sub>
	x <sub>1</sub>	x <sub>2</sub>	X <sub>0</sub>	X <sub>1</sub>	X <sub>2</sub>	X <sub>12</sub>					
1	150	5	1	-1	-1	1	569	574	571.5	12.5	571.5
2	150	10	1	-1	1	-1	550	563	556.5	84.5	556.5
3	200	5	1	1	-1	-1	570	580	575	50	575
4	200	10	1	1	1	1	529	536	532.5	24.5	532.5
Дополнительный опыт											
5	187.5	8.75	1	0.5	0.5	0.25	548	563	555.5	112.5	547.41

Таблица 8. Критерий Стьюдента для построения модели зависимости длины отражённой волны от времени и мощности. Оценка значимости факторов (10), (13), (14)

i	b <sub>i</sub>	t <sub>i</sub>	t <sub>кр</sub>	Значимость
0	558.875	241.41	2.13	Значим
1	-5.125	2.21		Значим
2	-14.375	6.21		Значим
12	-6.875	2.97		Значим

Таблица 9. Сводная таблица по моделированию длины отражённой волны от времени и мощности

Критерий	Значение
Экспериментальный критерий Кохрена (7)	0.49
Критический критерий Кохрена (8)	0.91
Дисперсия воспроизводимости (11)	42.875
Дисперсия адекватности (16)	131.02
Экспериментальный критерий Фишера (19)	3.06
Критический критерий Фишера (20)	6.61

$$Y_\lambda = 558.875 - 5.125X_1 - 14.375X_2 - 6.875X_1X_2 \quad (22)$$

### 1.5.2 Интенсивность отражения

Таблица 10. Матрица планирования эксперимента по определению зависимости значения интенсивности отражения от времени и мощности

№	P	t	k	Матрица планирования эксперимента								
				Y – R, %								
				1	2							
x <sub>1</sub>	x <sub>2</sub>	X <sub>0</sub>	X <sub>1</sub>	X <sub>2</sub>	X <sub>12</sub>	Y <sub>1</sub>	Y <sub>2</sub>	Y <sub>ср</sub>	S <sub>i</sub> <sup>2</sup>	Y <sub>мод</sub>		
1	150	5	1	-1	-1	1	17.5	20.3	18.9	3.92	18.9	
2	150	10	1	-1	1	-1	7.32	9.05	8.185	1.496	8.185	
3	200	5	1	1	-1	-1	12.2	11.9	12.05	0.045	12.05	
4	200	10	1	1	1	1	4.75	6.8	5.775	2.101	5.775	
Дополнительный опыт												
5	187.5	8.75	1	0.5	0.5	0.25	8.2	7.6	7.9	0.18	8.22	

Таблица 11. Критерий Стьюдента для построения модели зависимости интенсивности отражения от времени и мощности. Оценка значимости факторов (10), (13), (14)

i	b <sub>i</sub>	t <sub>i</sub>	t <sub>кр</sub>	Значимость
0	11.22	23.09	2.13	Значим
1	-2.315	4.76		Значим
2	-4.2475	8.73		Значим
12	1.11	2.28		Значим

Таблица 12. Сводная таблица по моделированию зависимости значений интенсивности отражения от времени и мощности

Критерий	Значение
Экспериментальный критерий Кохрена (7)	0.52
Критический критерий Кохрена (8)	0.91
Дисперсия воспроизводимости (11)	1.89
Дисперсия адекватности (16)	0.21
Экспериментальный критерий Фишера (19)	0.11
Критический критерий Фишера (20)	6.61

$$Y_R = 11.122 - 2.315X_1 - 4.2475X_2 + 1.11X_1X_2 \quad (23)$$

### 1.5.3 Результаты по 3 экспериментам

В результате проведённого анализа и полученных математических моделей (15), (22), (23) имеем 3 математические модели, дающие полное представление о параметрах плёнки:

$$\begin{aligned} Y_{шз3} &= 49.625 - 2.375X_1 - 5.375X_2 - 3.875X_1X_2 \\ Y_\lambda &= 558.875 - 5.125X_1 - 14.375X_2 - 6.875X_1X_2 \\ Y_R &= 11.122 - 2.315X_1 - 4.2475X_2 + 1.11X_1X_2 \end{aligned} \quad (24)$$

## 1.6 Анализ результатов эксперимента и выводы

Используя математические модели (24) в программном пакете GNU Octave [4] были построены поверхности зависимости трёх входных данных от времени и мощности (см. рис. 3, рис. 4, рис. 5).

Сравнение дисперсии адекватности и дисперсии воспроизводимости показало, что полученная математическая модель адекватно описывает влияние мощности магнетрона и времени распыления на оцениваемые параметры.

Факторы, оказывающие наибольшее влияние на

— Ширину ФЗЗ – время;

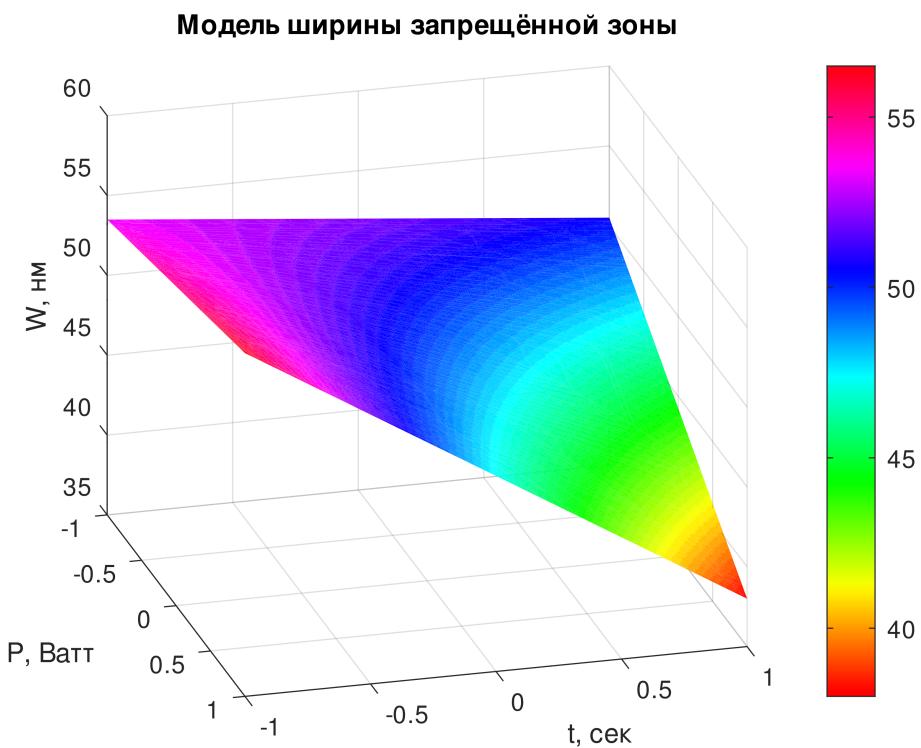


Рис. 3. Зависимость ширины фотонной запрещённой зоны от времени и мощности распыления

- Длину отражённой волны – взаимодействие факторов;
- Процент отражённого света – время распыления.

При увеличении времени нанесения уменьшается длина отражаемой волны и степень отражения.

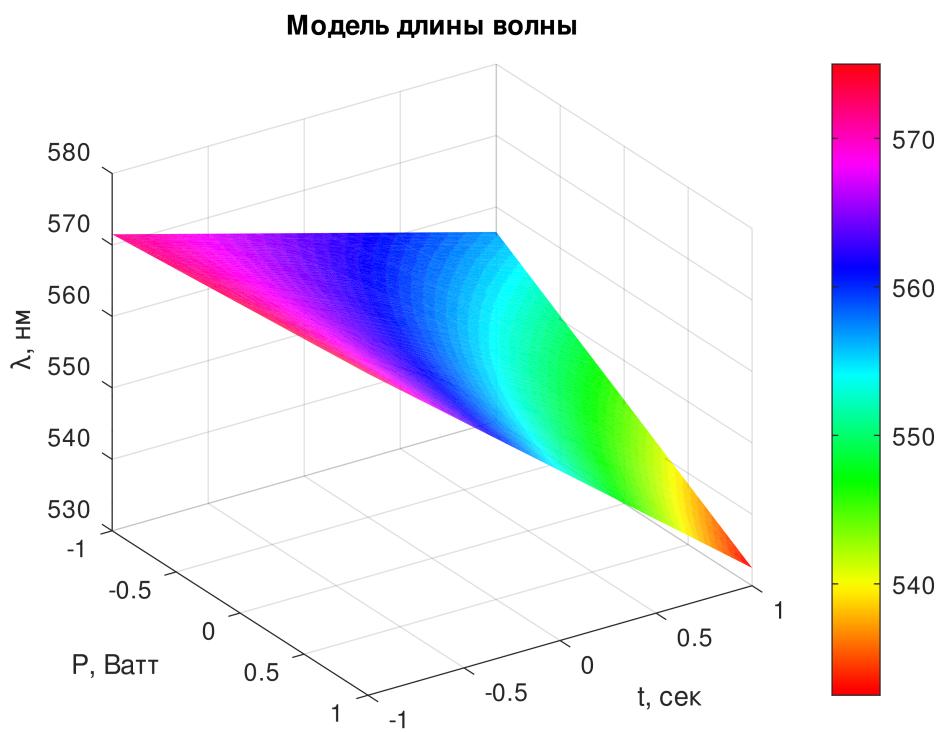


Рис. 4. Зависимость длины отраженной волны от времени и мощности распыления

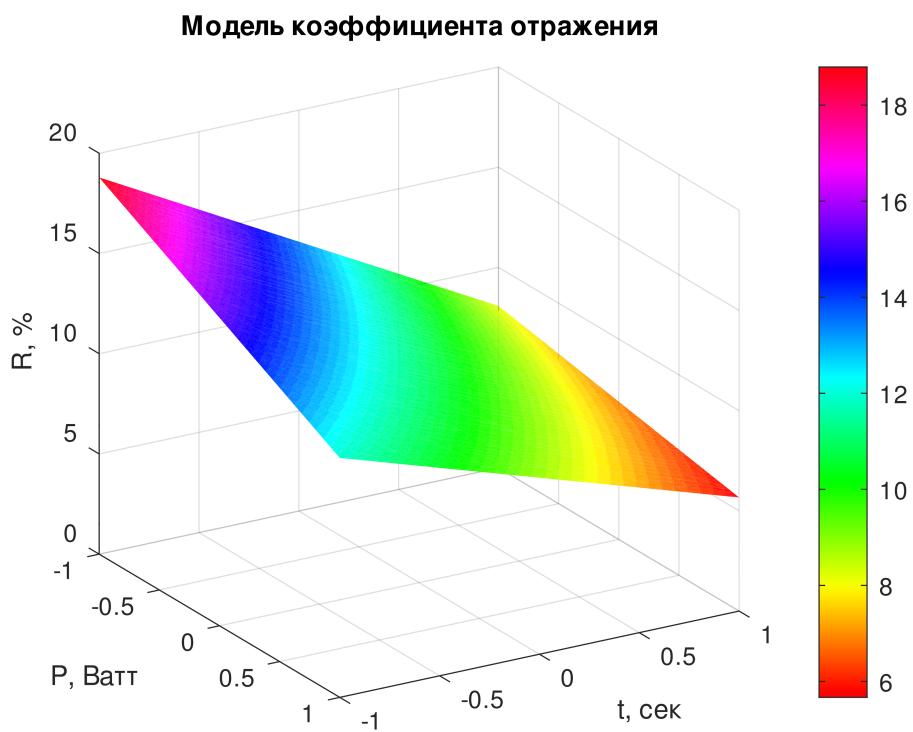


Рис. 5. Зависимость интенсивности отражения от времени и мощности распыления

## 2 Машинное обучение в Jupyter Notebook

Первым делом необходимо импортировать необходимые для обучения библиотеки

```
1 import numpy as np # for matrix operations
2 from sklearn.neural_network import MLPRegressor as mlp # main ml class
3 from matplotlib import pyplot as plt # для построения графиков
4 # show plot inside notebook
5 %matplotlib inline
6 from IPython.display import set_matplotlib_formats # hi-res plots
7 set_matplotlib_formats('pdf', 'png')
8 from joblib import dump, load # для загрузки и выгрузки моделей
9 plt.rcParams['figure.figsize'] = [10, 10]
```

### 2.1 Добавление данных

У нас имеются экспериментальные данные полнофакторного эксперимента по измерению характеристик фотонного кристалла в зависимости от времени распыления магнетрона и мощности на нём. В качестве входных данных выступают два массива: мощность на магнетроне X1 и время распыления X2

Выходные параметры – ширина запрещённой зоны, нм; длина отражённой волны, нм; степень отражения, %

```
1 #input
2 X0 = np.ones(4)
3 X1 = np.array([150, 150, 200, 200])
4 X2 = np.array([5, 10, 5, 10])
5
6 #output
7 Y1 = np.array([53.5, 50.5, 56.5, 38])
8 Y2 = np.array([571.5, 556.5, 575, 532.5])
9 Y3 = np.array([18.9, 8.185, 12.05, 5.775])
```

## 2.2 Нормализация данных

Для корректной работы нейросети данные необходимо нормализовать в пределах  $x \in [-1; 1]$

```
1 X1 = np.interp(X1, (X1.min(), X1.max()), (-1, +1))  
2 X2 = np.interp(X2, (X2.min(), X2.max()), (-1, +1))
```

## 2.3 Выбор оптимальных параметров для обучения

При обучении необходимо учитывать множество факторов, среди них:

- размерность сети;
- число скрытых слоёв;
- функция активации;
- оптимизатор.

### 2.3.1 Методы контроля

В качестве метода контроля за качеством обучения предлагается следующий алгоритм:

1. Взять заранее рассчитанные математические модели из §1.5.3 на стр. 16;
2. Подставить в них некие случайные значения, то есть получить отображение  $\mathbb{X} \rightarrow \mathbb{Y}$ ;
3. Использовать полученные массивы в качестве *test*-выборки для проверки качества полученной модели.

Создадим 4 варианта тестирований:

1.  $X_{matmod\_test}, y_{matmod\_test}$ , для удобства отображения предсказаний моделей с неизменными размерностями на входе и выходе;
2.  $X_{matmod\_test\_12}, y_{matmod\_test}$ , для удобства отображения предсказаний модели с двумя входными векторами и тремя выходными;

3.  $X_{matmod\_test\_full}, y_{matmod\_test\_full}$ , для валидации моделей с неизменными размерностями на входе и выходе;

4.  $X_{matmod\_test\_notfull}, y_{matmod\_test\_full}$ , для валидации модели с двумя входными векторами и тремя выходными;

```
1 # создадим 3 массива размерностью 1000 входные ( данные для валидации)
2 X0_matmod = np.ones(1000)
3 X1_matmod = np.linspace(-1,1,1000)
4 X2_matmod = np.linspace(-1,1,1000)
5
6 # создадим массивы небольшой размерности для удобства сверки
7 X0_matmod_test = np.ones(4) # генерация 4 единиц в массив
8 X1_matmod_test = np.random.choice(X1_matmod, size = 4, replace = False) #
    генерация 4 значений без повторений
9 X2_matmod_test = np.random.choice(X2_matmod, size = 4, replace = False)
10 X_matmod_test = np.transpose(np.array([X0_matmod_test, X1_matmod_test,
    X2_matmod_test])) # сборка массива и транспонирование
11 X_matmod_test_12 = np.transpose(np.array([X1_matmod_test, X2_matmod_test])) #
    специальный массив без X0 для проверки необходимости в X0
12
13 # генерация выходных данных из рассчитанной модели для теста
14 y1_matmod_test = 49.625 - 2.375 * X1_matmod_test - 5.375 * X2_matmod_test -
    3.875 * X1_matmod_test * X2_matmod_test # вычисляем Y при помощи X
15 y2_matmod_test = 558.875 - 5.125 * X1_matmod_test - 14.375 * X2_matmod_test -
    6.875 * X1_matmod_test * X2_matmod_test
16 y3_matmod_test = 11.122 - 2.315 * X1_matmod_test - 4.2475 * X2_matmod_test +
    1.11 * X1_matmod_test * X2_matmod_test
17 y_matmod_test = np.transpose(np.array([y1_matmod_test, y2_matmod_test,
    y3_matmod_test]))
18
19 # получаем на выходе три массива
20 print(X_matmod_test)
21 print(X_matmod_test_12)
22 print(y_matmod_test)
23
24 # генерируем большой массив для более точной сверки
25 X_matmod_test_full = np.transpose(np.array([X0_matmod, X1_matmod, X2_matmod])) #
    # сборка большого массива и транспонирование
26 X_matmod_test_notfull = np.transpose(np.array([X1_matmod, X2_matmod])) #
    специальный массив без X0 для проверки необходимости в X0
```

```

27 y1_matmod_test_full = 49.625 - 2.375 * X1_matmod - 5.375 * X2_matmod - 3.875 *
    X1_matmod * X2_matmod # вычисляем Y при помощи X
28 y2_matmod_test_full = 558.875 - 5.125 * X1_matmod - 14.375 * X2_matmod - 6.875
    * X1_matmod * X2_matmod
29 y3_matmod_test_full = 11.122 - 2.315 * X1_matmod - 4.2475 * X2_matmod + 1.11 *
    X1_matmod * X2_matmod
30 y_matmod_test_full = np.transpose(np.array([y1_matmod_test_full,
    y2_matmod_test_full, y3_matmod_test_full]))

```

```

1 [[ 1.          0.48348348  0.33533534]
2 [ 1.         -0.20920921 -0.74174174]
3 [ 1.          0.73773774 -0.36536537]
4 [ 1.          0.04904905 -0.28328328]]
5 [[ 0.48348348  0.33533534]
6 [-0.20920921 -0.74174174]
7 [ 0.73773774 -0.36536537]
8 [ 0.04904905 -0.28328328]]
9 [[ 46.04604905 550.46206417   8.7583622 ]
10 [ 53.50741432 569.54287771  14.92911628]
11 [ 50.88119401 562.19933497  10.66683289]
12 [ 51.08499841 562.7913474   12.196274 ]]

```

## 2.4 Влияние размерности сети на качество модели

Создадим 2 массива  $X$ : с 2 и 3 элементами соответственно. Сделано это для потому, что нейронной сети очень сложно сделать отображение из меньшей размерности в большую. Вектор  $X_0$  является ни чем иным, как вектором-заглушкой, не вносящим изменения в модель, но искусственно раздувающий размерность.

```

1 X_012_train = np.array([X0,X1,X2]) # вектор размерности 3
2 X_12_train = np.array([X1,X2]) # вектор размерности 2
3 X_012_train

```

```

1 array([[ 1.,  1.,  1.,  1.],
2 [-1., -1.,  1.,  1.],
3 [-1.,  1., -1.,  1.]])

```

Создадим 1 массив  $\mathbb{Y}$ , поскольку нас интересует 3 выходных параметра.

Размерность этого вектора мы варьировать не будем

```
1 Y_train=np.array([Y1,Y2,Y3])
2 Y_train
1 array([[ 53.5   ,  50.5   ,  56.5   ,  38.    ],
2 [571.5   , 556.5   , 575.    , 532.5   ],
3 [ 18.9   ,   8.185,  12.05 ,  5.775]])
```

## 2.4.1 Транспортирование массивов

Получившиеся выше массивы не совсем подходят для обработки. Нам необходимо отображение комбинаций  $\mathbb{X}$  в комбинации  $\mathbb{Y}$ . Для этого транспортируем обе матрицы.

```
1 X_012_train = np.transpose(X_012_train)
2 X_12_train = np.transpose(X_12_train)
3 X_012_train
1 array([[ 1., -1., -1.],
2 [ 1., -1.,  1.],
3 [ 1.,  1., -1.],
4 [ 1.,  1.,  1.]])
1 Y_train = np.transpose(Y_train)
2 Y_train
1 array([[ 53.5   ,  571.5   ,  18.9   ],
2 [ 50.5   , 556.5   ,   8.185],
3 [ 56.5   , 575.    ,  12.05 ],
4 [ 38.    , 532.5   ,  5.775]])
```

## 2.4.2 Создание архитектуры модели

Создадим 2 модели, отвечающие за размерности векторов. Оставим все значения по умолчанию кроме входных параметров.

```
1 model_012 = mlp(random_state=1,max_iter=1000000)
2 model_12 = mlp(random_state=1,max_iter=1000000)
```

Проведём обучение моделей на заранее подготовленных выборках

```
1 model_012.fit(X_012_train, Y_train)
2 model_12.fit(X_12_train, Y_train)

1 MLPRegressor(max_iter=1000000, random_state=1)
```

Проведём проверку качества работы моделей при помощи заранее заготовленных тестов. Функция  $score$  – функция  $\mathbb{R}^2$ . Наилучшая возможная оценка – 1, и она может быть отрицательной (потому что модель может быть произвольно хуже). Постоянная модель, которая всегда предсказывает ожидаемое значение  $y$ , игнорируя входные характеристики, получит оценку 0.

$$R = \left(1 - \frac{u}{v}\right) \quad (25)$$

—  $u$  – остаточная сумма квадратов. Метод оценки разницы между данными и оценочной моделью. Чем меньше разница, тем лучше оценка. По существу измеряет разброс ошибок моделирования. Другими словами, он показывает, как изменение зависимой переменной в регрессионной модели не может быть объяснено с помощью модели. Как правило, более низкая остаточная сумма квадратов указывает на то, что регрессионная модель может лучше объяснить данные, в то время как более высокая остаточная сумма квадратов указывает на то, что модель плохо объясняет данные.

$$u = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (26)$$

- $y_i$  – действительное значение;
- $\hat{y}_i$  – значение, полученное моделью.

—  $v$  – общая сумма квадратов. Представляет собой отклонение значений зависимой переменной от выборочного среднего значения зависимой переменной. По сути, общая сумма квадратов количественно определяет общую вариацию в выборке. Его можно определить по следующей фор-

мудле:

$$v = \sum_{i=1}^n (y_i - \bar{y}_i)^2 \quad (27)$$

- $y_i$  – действительное значение;
- $\bar{y}_i$  – среднее значение.

```
1 print("Модель 012")
2 print(model_012.score(X_matmod_test_full,y_matmod_test_full))
3 print(model_012.predict(X_matmod_test))
4
5 print("Модель 12")
6 print(model_12.score(X_matmod_test_notfull,y_matmod_test_full))
7 print(model_12.predict(X_matmod_test_12))
```

```
1 Модель 012
2 0.8135462211671235
3 [[ 43.40571923 543.68112196   9.65241689]
4 [ 50.90158838 562.20556561  15.87171834]
5 [ 46.43032235 552.39883876  12.27512113]
6 [ 47.8070884  554.49364847  13.29834493]]
7 Модель 12
8 -16.94187285805916
9 [[ 36.32865331 440.61778227   5.53561853]
10 [ 48.04352244 484.47184626  13.96828786]
11 [ 41.28359463 459.47902463   8.88703558]
12 [ 43.18840615 465.67895223  10.44771188]]
```

Как видно, модель с раздутой размерностью показала себя лучше. Построим график сходимости каждой из моделей

```
1 y_curve_012 = model_012.loss_curve_
2 x_curve_012 = [i for i in range(1,len(y_curve_012)+1)]
3 print(f'Модель 012 сошлась за {len(x_curve_012)} итераций, потери составили {model_012.best_loss_}')
4
5 y_curve_12 = model_12.loss_curve_
6 x_curve_12 = [i for i in range(1,len(y_curve_12)+1)]
7 print(f'Модель 12 сошлась за {len(x_curve_12)} итераций, потери составили {model_12.best_loss_}')
8
9 plt.plot(x_curve_12, y_curve_12,label='Модель 12')
```

```

10 plt.plot(x_curve_012, y_curve_012,label='Модель 012')
11 plt.xlabel("Число итераций")
12 plt.ylabel("Loss")
13 plt.legend()
14 plt.title("Функция потерь двух архитектур моделей")

```

```

1 Модель 012 сошлась за 7864 итераций, потери составили 0.05247663242357793
2 Модель 12 сошлась за 4114 итераций, потери составили 0.03380976458075128

```

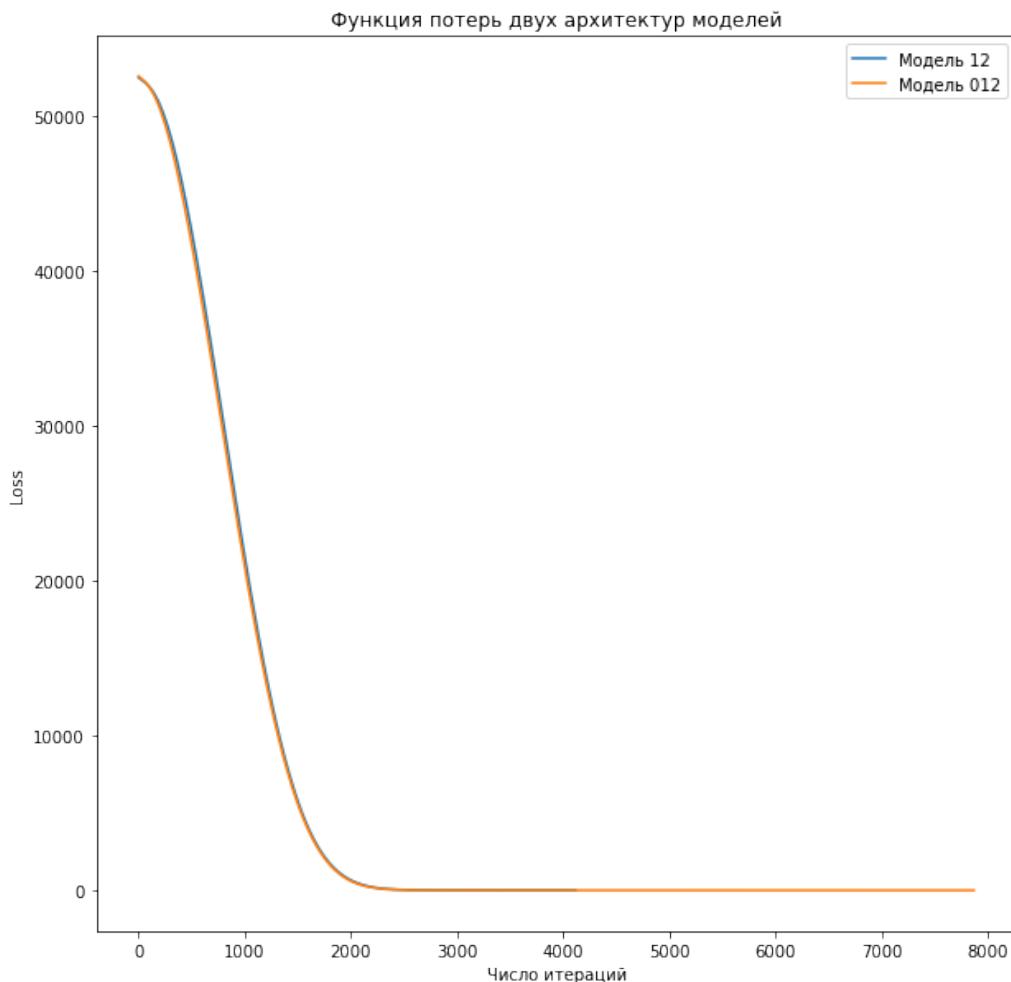


Рис. 6. Скорость обучения архитектур с различным числом нейронов

### 2.4.3 Выводы

Модель 012 сходится быстрее и имеет лучшую оценку (0.81 по сравнению с -16). Необходимо брать модель размерности не меньше, чем желаемый результат.

## 2.5 Влияние числа скрытых слоёв сети на качество модели

Как мы убедились в прошлом разделе, модель должна иметь размерность на входе не ниже размерности на выходе, иначе потери  $R^2$  слишком высоки. Посмотрим теперь что будет, если варьировать количество скрытых слоёв модели (например, от 1 до 100). Необходимо помнить, что слишком сложная модель может переобучиться на небольшой выборке. Будем сохранять модели в отдельные файлы, поскольку их обучение занимает длительное время

```
1 loss = []
2 points = []
3 iters = []
4 for layers in range(2,101):
5     varymodel = mlp(random_state=1,max_iter=1000000,hidden_layer_sizes=(layers))
6     varymodel.fit(X_012_train, Y_train)
7     loss.append(varymodel.best_loss_)
8     points.append(varymodel.loss_curve_)
9     iters.append(varymodel.n_iter_)
10    #dump(varymodel,f'models/layers/{layers}.joblib')
11    print(layers)
```

Для того, чтобы не проводить многократную тренировку модели, проведём тренировку один раз, а в остальные будем загружать модели из готовых файлов.

```
1 loss = []
2 points = []
3 iters = []
4 scores = []
5 for layers in range(2,101):
6     varymodel = load(f'models/layers/{layers}.joblib')
7     loss.append(varymodel.best_loss_)
8     points.append(varymodel.loss_curve_)
9     iters.append(varymodel.n_iter_)
10    scores.append(varymodel.score(X_matmod_test_full,y_matmod_test_full))

1 loss[5] = 0 # выборочная ошибка, занулим её
2 scores[5] = 0

1 layers = list(range(2,101))
```

```

2 plt.plot(layers, loss)
3 plt.xlabel("Число слоёв")
4 plt.ylabel("Loss")
5 plt.title("Влияние числа слоёв на потери при обучении")

```

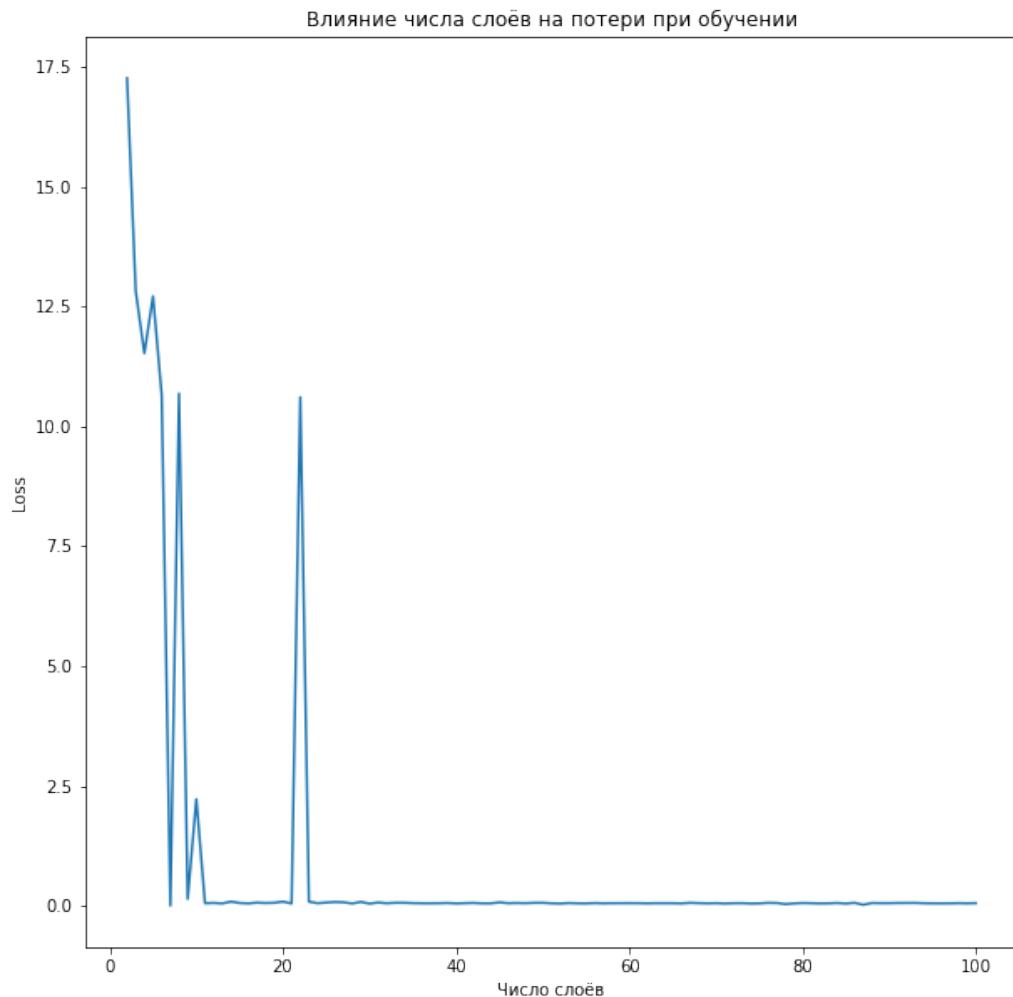


Рис. 7. Влияние числа слоёв на потери при обучении

```

1 for i in range (len(iters)):
2     #print(i)
3     if (i>10 and i % 5 != 0):
4         continue
5     x = list(range(1,iters[i]+1))
6     y = points[i]
7     plt.plot(x, y, label=f'Слой {i+2}')
8     plt.xlabel("Число итераций")
9     plt.ylabel("Потери")
10    plt.title("Влияние числа слоёв на качество модели")
11    plt.legend()

```

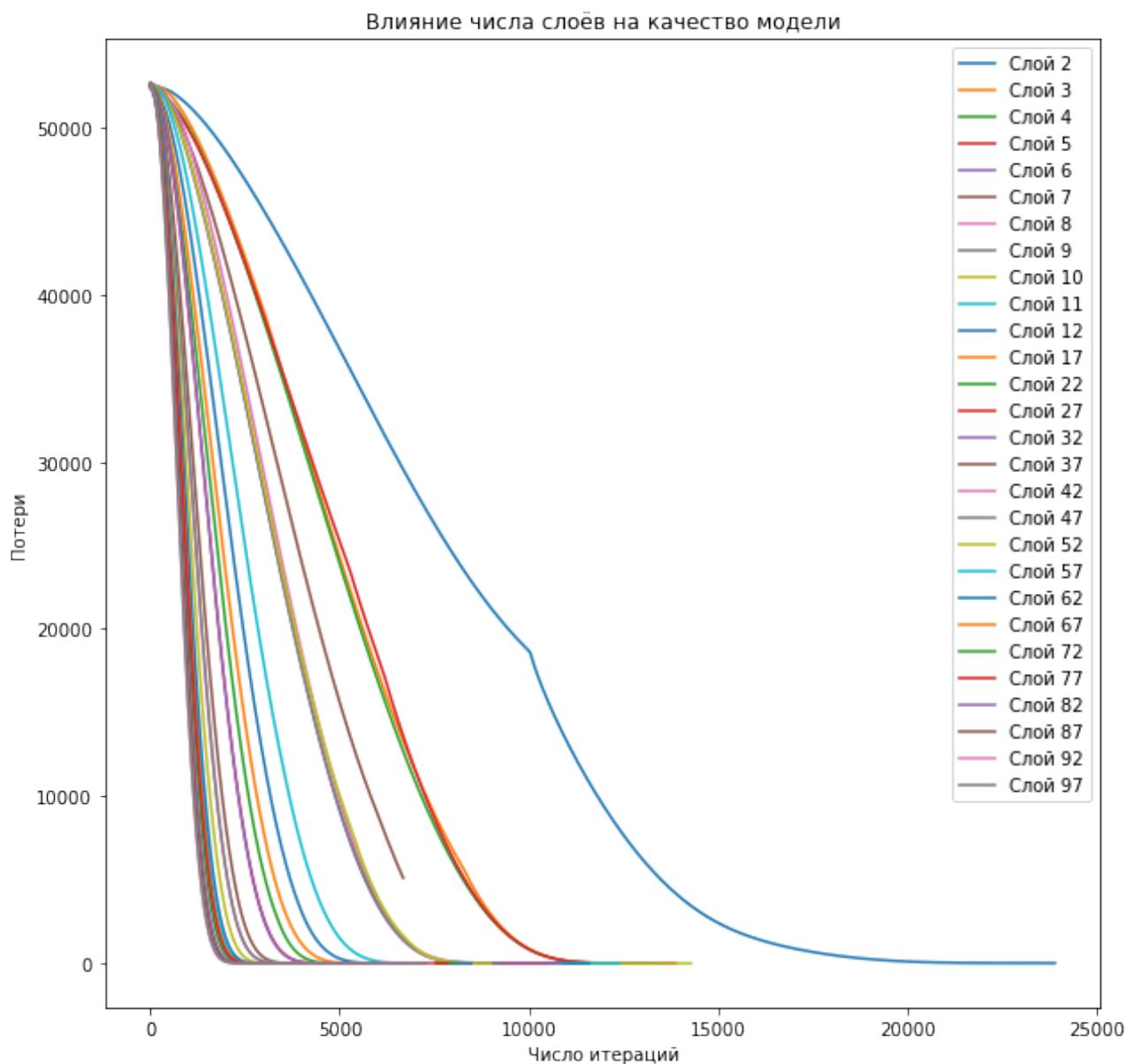


Рис. 8. Влияние числа слоёв на скорость сходимости

Самым главным параметром при обучении будет являться не скорость сходимости модели, а её оценка. Проведём оценку модели

```

1 plt.plot(list(range(2,101)), scores)
2 plt.xlabel("Число слоёв")
3 plt.ylabel("Оценка модели")
4 plt.title("Влияние числа слоёв на оценку модели")

1 Text(0.5, 1.0, 'Влияние' ' числа слоёв на оценку модели')

```

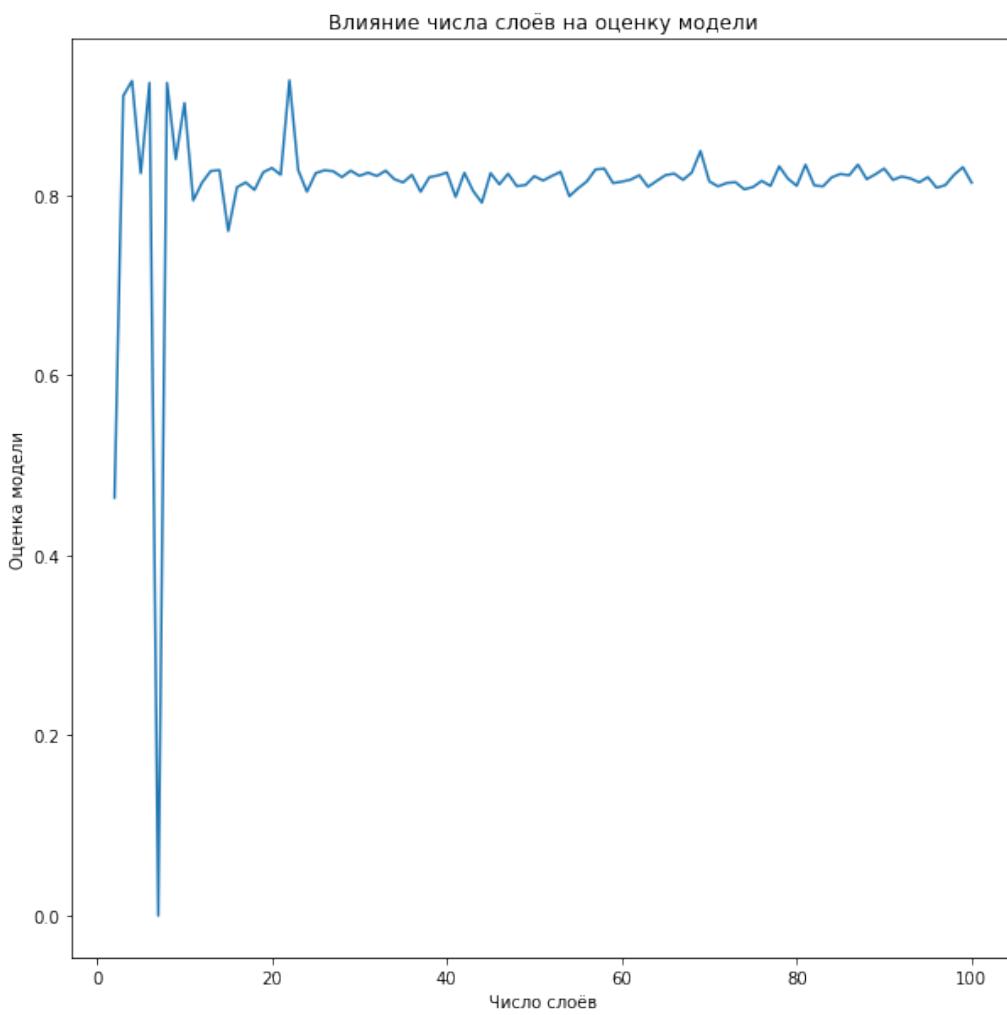


Рис. 9. png

Очевидно, что нас интересует всё до десяти слоёв, дальше модель начинает переобучаться (резкое падение оценки – признак переобучения)

```

1 plt.plot(list(range(2,10)), scores[:8])
2 plt.xlabel("Число слоёв")
3 plt.ylabel("Оценка модели")
4 plt.title("Влияние числа слоёв на оценку модели")
5 for i in range(len(scores[:8])):
6     print(f'Число слоёв {i+2}, оценка модели {scores[i]}')

```

```

1 Число слоёв 2, оценка модели 0.4639444511330419
2 Число слоёв 3, оценка модели 0.9100311255101875
3 Число слоёв 4, оценка модели 0.9265799685143974
4 Число слоёв 5, оценка модели 0.8241257764470432
5 Число слоёв 6, оценка модели 0.924471640369974
6 Число слоёв 7, оценка модели 0
7 Число слоёв 8, оценка модели 0.9244552323152719

```

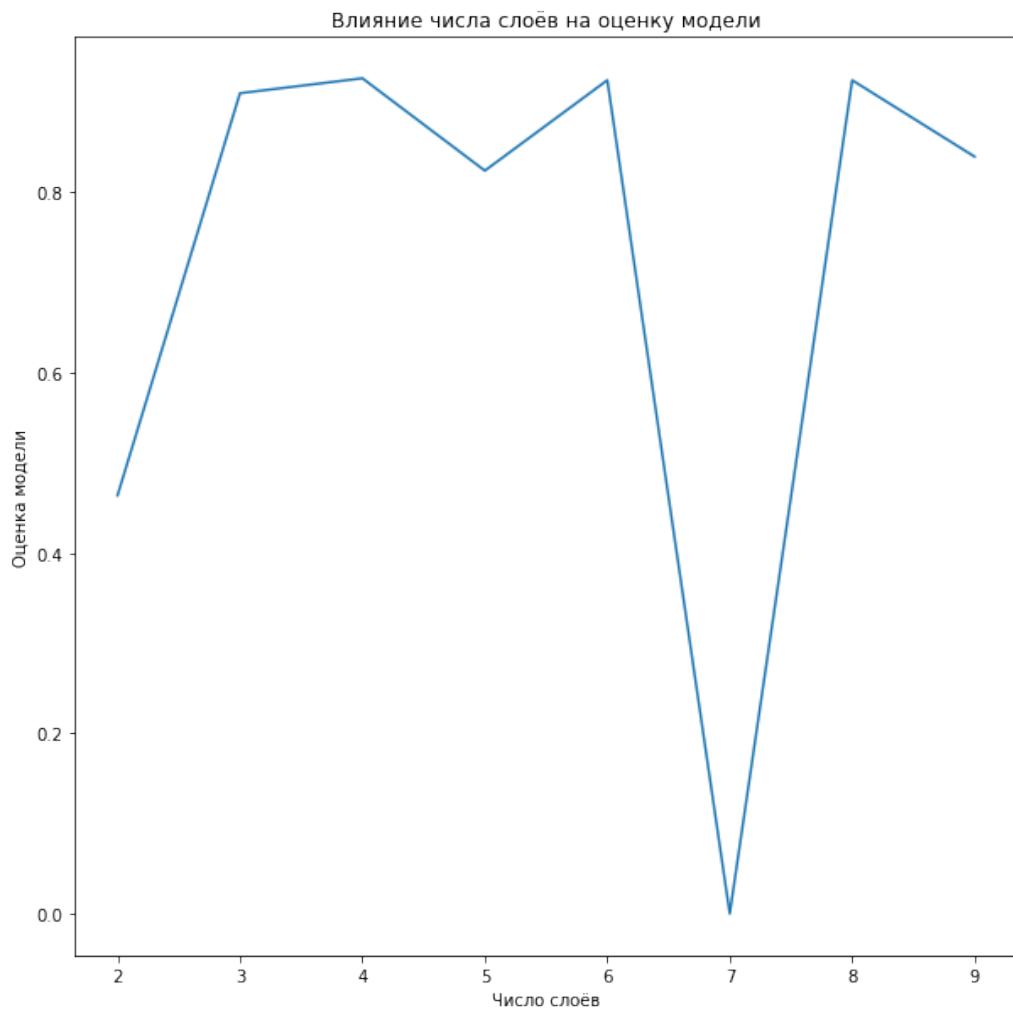


Рис. 10. png

### 2.5.1 Выводы

Из графика видно, что лучше всего нам подходит модель с числом слоёв, равным 8 (далее наблюдается переобучение, нежелательное явление).

## 2.6 Влияние функции активации на результаты модели

Библиотека *scikit-learn* предоставляет доступ к 4 функциям активации:

1. *identity* – функция тождества. Безоперационная активация, полезная

для реализации линейного узкого места

$$f(x) = x \quad (28)$$

2. *logistic* – сигмоидальная функция

$$f(x) = \frac{1}{1 + e^{-x}} \quad (29)$$

3. *tanh* – гиперболический тангенс

$$f(x) = \tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (30)$$

4. *relu* – линейный выпрямитель

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (31)$$

Обучим 4 модели при помощи каждой из функций активаций

```
1 scores_activation = []
2 model_identity = mlp(random_state=1,max_iter=2000000,hidden_layer_sizes=(8),\
3 activation='identity')
4 model_identity.fit(X_012_train, Y_train)
5 scores_activation.append(model_identity.score(X_matmod_test_full,
6 y_matmod_test_full))
7
8 model_logistic = mlp(random_state=1,max_iter=2000000,hidden_layer_sizes=(8),\
9 activation='logistic')
10 model_logistic.fit(X_012_train, Y_train)
11 scores_activation.append(model_logistic.score(X_matmod_test_full,
12 y_matmod_test_full))
13
14 model_tanh = mlp(random_state=1,max_iter=2000000,hidden_layer_sizes=(8),\
15 activation='tanh')
16 model_tanh.fit(X_012_train, Y_train)
17 scores_activation.append(model_tanh.score(X_matmod_test_full,
18 y_matmod_test_full))
```

```
18 activation='relu')
19 model_relu.fit(X_012_train, Y_train)
20 scores_activation.append(model_relu.score(X_matmod_test_full,
    y_matmod_test_full))
```

```
1 x = [1, 2, 3, 4]
2 y = scores_activation
3 my_xticks = ['identity','logistic','tanh','relu']
4 plt.xticks(x, my_xticks)
5 plt.scatter(x,y)
6 plt.xlabel("Способ обучения")
7 plt.ylabel("Оценка модели")
8 plt.title("Влияние числа слоёв на оценку модели")
9 print(scores_activation)
```

```
1 [0.9275591949671171, -0.03702885533324888, -0.03702338056124536,
0.9244552323152719]
```

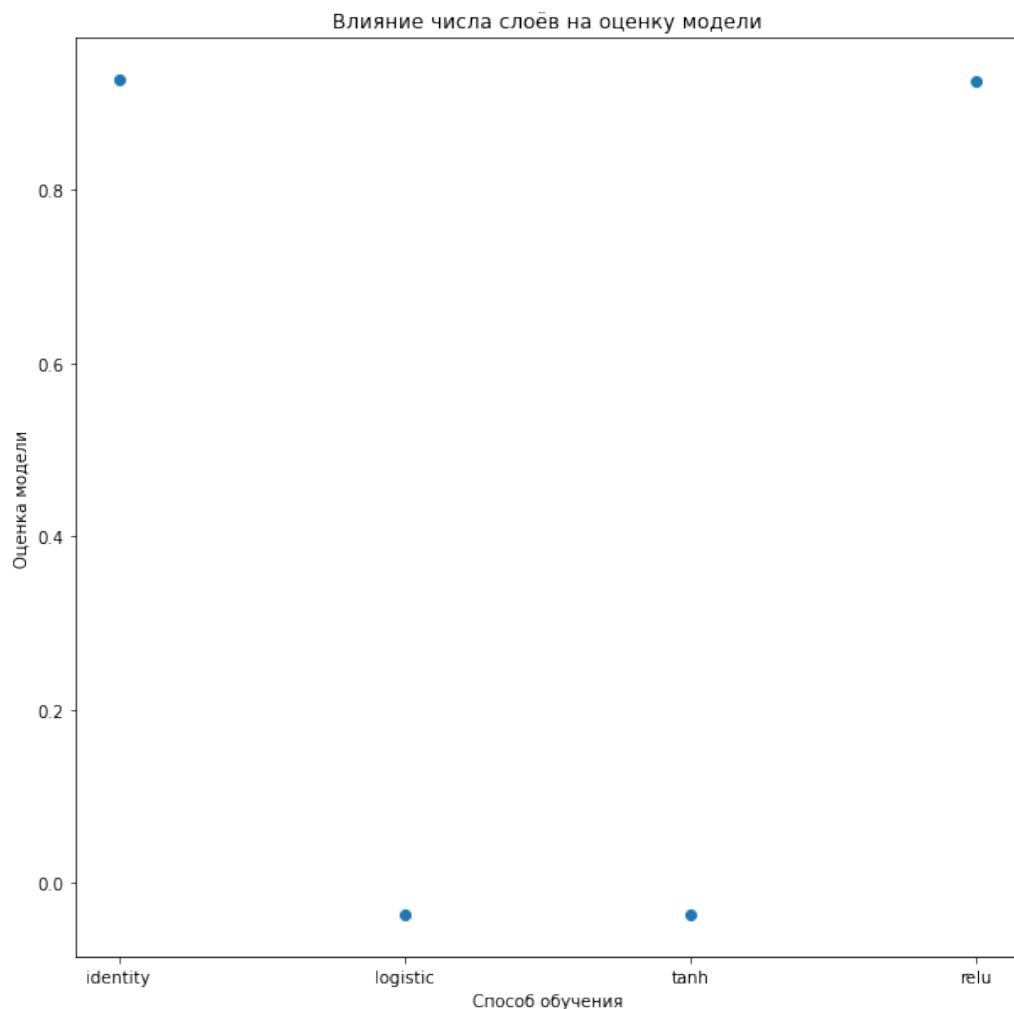


Рис. 11. Влияние числа слоёв на оценку модели

## 2.6.1 Выводы

Из графика видно, что наилучший результат показывает отсутствие функции активации (*identity*)

## 2.7 Влияние солвера на результаты модели

Под солвером понимается один из алгоритмов для обучения нейросети. Библиотека *scikit-learn* предоставляет доступ к 3 алгоритмам обучения:

1. *lbfgs* – алгоритм Бройдена - Флетчера - Гольдфарба - Шанно, рекомендуется для небольших датасетов;
2. *sgd* – стохастический градиентный спуск, один из классических методов обучения;
3. *adam* – метод адаптивной скорости обучения, используется по умолчанию.

Обучим модель с использованием этих алгоритмов

```
1 algorithms = []
2 model_lbfgs = mlp(random_state=1,max_iter=2000000,hidden_layer_sizes=(8),\
3 activation='identity',solver='lbfgs')
4 model_lbfgs.fit(X_012_train, Y_train)
5 algorithms.append(model_lbfgs.score(X_matmod_test_full,y_matmod_test_full))
6
7 model_adam = mlp(random_state=1,max_iter=2000000,hidden_layer_sizes=(8),\
8 activation='identity',solver='adam')
9 model_adam.fit(X_012_train, Y_train)
10 algorithms.append(model_adam.score(X_matmod_test_full,y_matmod_test_full))
```

При обучении модели методом стохастического градиентного спуска нейросеть не смогла добиться оптимальных параметров и обучение было аварийно завершено

```
1 x = [1, 2]
2 y = algorithms
3 my_xticks = ['lbfgs', 'adam']
4 plt.xticks(x, my_xticks)
5 plt.scatter(x,y)
```

```
6 plt.xlabel("Оптимизационный алгоритм")
7 plt.ylabel("Оценка модели")
8 plt.title("Влияние оптимизационного алгоритма на оценку модели")
9 print(algorithms)

1 [0.9247783928754306, 0.9275591949671171]
```

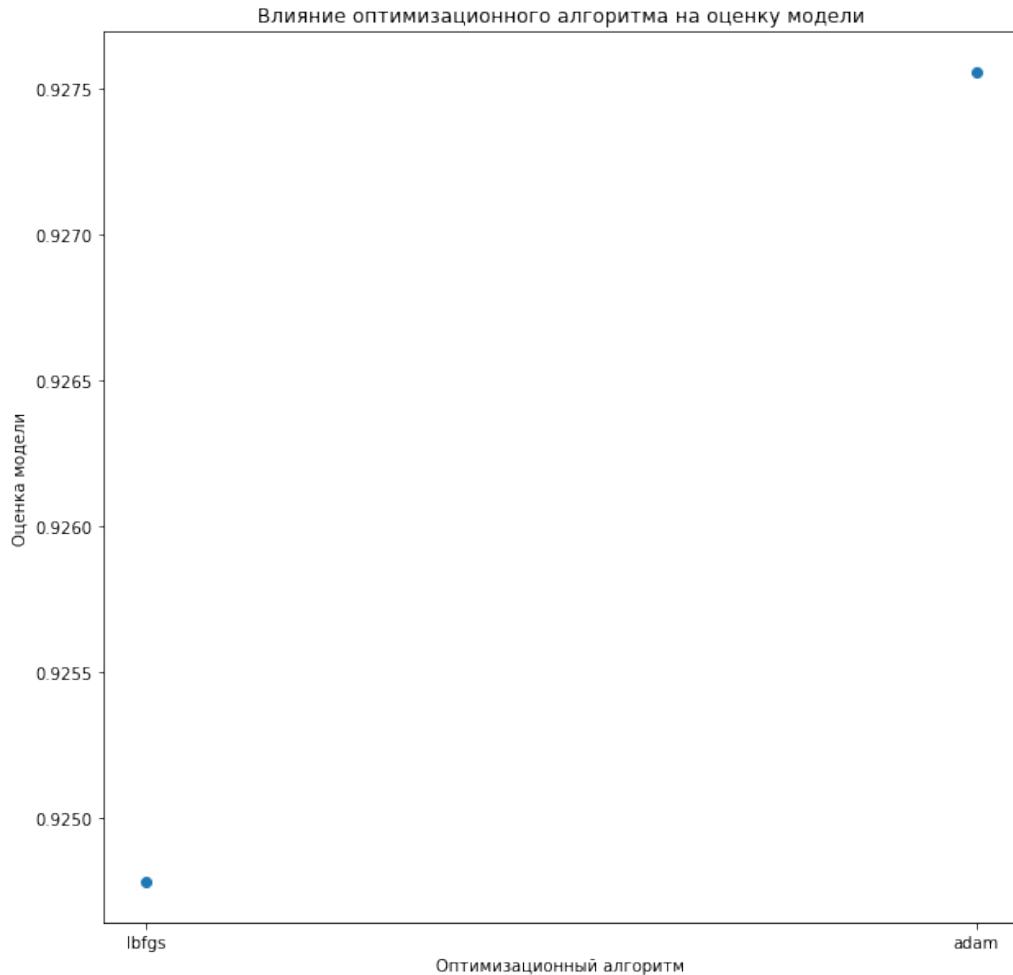


Рис. 12. Влияние оптимизационного алгоритма на оценку модели

### 2.7.1 Вывод

Влияние солвера не настолько велико, но мы отдадим предпочтение *Adam*.

## 2.8 Проверка полученной модели

После проведения экспериментов с моделями, попробуем проверить полученную модель на всей экспериментальной выборке. Обучим верную модель

```

1 final_model = mlp(random_state=1,max_iter=2000000,hidden_layer_sizes=(8),\
2 activation='identity',solver='adam')
3 final_model.fit(X_012_train, Y_train)

1 MLPRegressor(activation='identity', hidden_layer_sizes=8, max_iter=2000000,
2 random_state=1)

1 final_model.score(X_matmod_test_full,y_matmod_test_full)

1 0.9275591949671171

```

Наша модель набрала 0.92 из 1, что является хорошим результатом при обучении на 4 входных данных

## 2.9 Получение результатов из модели

После обучения модели встаёт вопрос: как получать из неё результат. Для этого в библиотеке *scikit-learn* в классе *MLPRegressor* метод *predict*. Попробуем получить из модели какое-нибудь предсказание, например, для точки  $X = [1, 0, 0]$ , что соответствует времени напыления  $t = 7.5$  минут при мощности  $P = 175$  Вт

```

1 [[a,b,c]] = final_model.predict([[1,0,0]])
2 print(f'''Данные по модели
3 Ширина запрещённой зоны {round(a,2)} нм
4 Длина отражённой волны {round(b,2)} нм
5 Процент отражённого света {round(c,2)} %''')

1 Данные по модели
2 Ширина запрещённой зоны 49.63 нм
3 Длина отражённой волны 558.62 нм
4 Процент отражённого света 11.23 %

```

Реальные данные (вычисленные по модели):

- Ширина запрещённой зоны 49.625 нм
- Длина отражённой волны 558.875 нм
- Процент отражённого света 11.122 %

Отклонения составили:

$$\delta S = \frac{S_{\text{real}} - S_{\text{model}}}{S_{\text{real}}} \cdot 100\% = \frac{49.625 - 49.63}{49.625} \cdot 100\% = -0.01\% \quad (32)$$

$$\delta \lambda = \frac{\lambda_{\text{real}} - \lambda_{\text{model}}}{\lambda_{\text{real}}} \cdot 100\% = \frac{558.675 - 558.62}{558.75} \cdot 100\% = 0.0098\% \quad (33)$$

$$\delta R = \frac{R_{\text{real}} - R_{\text{model}}}{R_{\text{real}}} \cdot 100\% = \frac{11.122 - 11.23}{11.122} \cdot 100\% = -0.97\% \quad (34)$$

Отклонения от матмодели получились меньше одного процента

```
1 print(f'Предсказание модели {final_model.predict(X_matmod_test)}')
2 print(f'Реальные значения {y_matmod_test}')
```

```
1 Предсказание модели [[ 46.67454687 551.32429405   8.68390614]
2 [ 54.10855985 570.35738101  14.86237024]
3 [ 49.83670087 560.09384672  11.07154185]
4 [ 51.03116084 562.44344424  12.31720136]]
5 Реальные значения [[ 46.04604905 550.46206417   8.7583622 ]
6 [ 53.50741432 569.54287771  14.92911628]
7 [ 50.88119401 562.19933497  10.66683289]
8 [ 51.08499841 562.7913474   12.196274 ]]
```

Сохраним полученную модель

```
1 dump(final_model,f'models/final_model.joblib')
```

```
1 ['models/final_model.joblib']
```

## **ЗАКЛЮЧЕНИЕ**

В ходе работы было проведено комплексное исследование обработки результатов экспериментов при помощи нейросетей.

Разработана нейросетевая структура для предсказания результатов последующих экспериментов точностью 92%.

Проанализировано влияние размерностей входных и выходных данных на качество модели. Для качественной работы нейросети размерность данных на входе должна быть не меньше, чем размерность данных на выходе.

Исследована зависимость точности сети от количества нейронов в скрытом слое. Для случая с четырьмя экспериментами и 3 входными данными достаточною оказалась нейросеть, содержащая 8 нейронов в скрытом слое.

Изучено влияние функции активации на результаты модели. При создании регрессионной модели функция активации не имеет смысла, что было подтверждено экспериментально.

Проверена зависимость параметров модели от алгоритма оптимизации. Алгоритм Adam показал себя лучше всех, поэтому и был выбран.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Галаганова Е. Н., Сотников Д. А. Получение нанокомпозитных структур при магнетронном осаждении материала на коллоидные пленки кремнезема и исследование их свойств // Наноиндустрия. — 2020. — Т. 13, S2. — С. 146—152. — ISSN 1993-8578, 2687-0282. — DOI: [10.22184/1993-8578.2020.13.2s.146.152](https://doi.org/10.22184/1993-8578.2020.13.2s.146.152). — URL: <https://elibrary.ru/item.asp?id=42712768> (дата обр. 25.11.2021).
2. Математическое Моделирование Процесса Формирования Композитной Структуры На Основе Фотонного Кристалла / Е. Н. Галаганова, М. К. Нгуен, Е. В. Панфилова, Д. А. Сотников //. — Общество с ограниченной ответственностью «Диона», 2020. — С. 73—76. — URL: <https://www.elibrary.ru/item.asp?id=43861616> (дата обр. 22.11.2021).
3. Панфилова Е. В. Техника эксперимента в электронике и наноэлектронике. Краткий конспект лекций: учебное пособие. — Москва : Издательство МГТУ им. Н. Э. Баумана, 2020. — 50 с.
4. GNU Octave version 6.3.0 manual: a high-level interactive language for numerical computations : manual / J. W. Eaton, D. Bateman, S. Hauberg, R. Wehbring. — 2021.
5. Community E. B. Jupyter Book. — Zenodo, 12.02.2020. — DOI: [10.5281/zenodo.4539666](https://doi.org/10.5281/zenodo.4539666). — URL: <https://zenodo.org/record/4539666> (дата обр. 23.12.2021).
6. Array programming with NumPy / C. R. Harris [и др.] // Nature. — 2020. — Сент. — Т. 585, вып. 7825, № 7825. — С. 357—362. — ISSN 1476-4687. — DOI: [10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2). — URL: <https://www.nature.com/articles/s41586-020-2649-2> (дата обр. 26.12.2021).

7. *Hunter J. D.* Matplotlib: A 2D Graphics Environment // Computing in Science Engineering. — 2007. — Май. — Т. 9, № 3. — С. 90—95. — ISSN 1558-366X. — DOI: [10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55).
8. Scikit-learn: Machine Learning in Python / F. Pedregosa [и др.] // Journal of Machine Learning Research. — 2011. — Т. 12, № 85. — С. 2825—2830. — ISSN 1533-7928. — URL: <http://jmlr.org/papers/v12/pedregosa11a.html> (дата обр. 23.12.2021).
9. *Perez F., Granger B. E.* IPython: A System for Interactive Scientific Computing // Computing in Science Engineering. — 2007. — Май. — Т. 9, № 3. — С. 21—29. — ISSN 1558-366X. — DOI: [10.1109/MCSE.2007.53](https://doi.org/10.1109/MCSE.2007.53).
10. API design for machine learning software: experiences from the scikit-learn project / L. Buitinck [и др.] // ECML PKDD Workshop: Languages for Data Mining and Machine Learning. — 2013. — С. 108—122. — URL: <https://arxiv.org/abs/1309.0238> (дата обр. 23.12.2021).