ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М. В. ЛОМОНОСОВА»

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА теории пластичности

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(ДИПЛОМНАЯ РАБОТА)
специалиста

ПРИМЕНЕНИЕ МЕТОДОВ АНАЛИЗА БОЛЬШИХ ДАННЫХ ДЛЯ ОЦЕНКИ ВЕЛИЧИНЫ И МЕСТОНАХОЖДЕНИЯ ПОВРЕЖДЕНИЯ В КОНСТРУКЦИИ

Выполнил студент
625 группы
Мищенко Александр Дмитриевич

_____
подпись студента

Научный руководитель: профессор Шешенин
Сергей Владимирович

_____
подпись научного руководителя

Москва
2018 год

Содержание

1 Введение

Дефекты в конструкциях являются основной причиной аварий и в то же время неизбежно возникают в процессе эксплуатации. Множество работ последних десятилетий было посвящено методологиям обнаружения дефектов на ранней стадии. Они позволяют проводить своевременную замену деталей конструкций без неприятных последствий. Сбор данных о состоянии конструкций и алгоритмы для обработки этих данных в совокупности носят название Structural Health Monitoring (SHM).

Задачи SHM принято разделять на четыре последовательные ступени [1]:

1. Определение наличия повреждения в конструкции.
2. Определение местонахождения повреждения в конструкции.
3. Оценка величины повреждения.
4. Прогнозирование оставшегося срока службы конструкции.

Эта работа сосредоточена на втором и третьем пунктах и затрагивает первый: они носят общее название *идентификации дефектов*. Последний пункт относится к так называемым задачам оценки ресурса конструкции и анализа усталостной долговечности.

Для идентификации дефектов применяются *недеструктивные методы*. Они позволяют определять параметры дефектов с минимально возможным вмешательством в исследуемый объект. Примеры недеструктивных методов [2]:

- Визуальный осмотр – невооружённым глазом или с помощью оптических приборов.
- Капиллярный контроль – полости поверхностных трещин заполняют специальными индикаторными веществами (пенетрантами), проникающими в трещины под действием сил капиллярности и повышающими контрастность дефектных участков.
- Магнитно-порошковая дефектоскопия. Принцип действия основан на создании поля рассеяния над дефектами детали с последующим выявлением их магнитной суспензией.
- Радиография – облучение ренгеновскими, α-, β-, γ- лучами и нейтронами в расчете на то, что трещины, раковины или включения инородного материала ослабляют лучи.
- Ультразвуковая дефектоскопия. Принцип действия основан на свойстве ультразвуковых колебаний отражаться от внутренних дефектов материала, проводящего эти колебания.

Однако перечисленные “ручные” методы не могут быть отнесены к SHM, поскольку не могут применяться автоматически без участия человека. Кроме того, в некоторых случаях они бесполезны: например, когда стоит задача мониторинга состояния мостов и других конструкций огромного размера.

Недеструктивные методы SHM заключаются в применении различных алгоритмов для решения *обратной задачи*. Разделение задач на прямые и обратные формулируется так [3]: прямая задача состоит в том, чтобы выяснить, как различные заданные повреждения влияют на параметры системы; тогда как обратная задача – более реалистичная – измерить параметры системы и по ним предсказать наличие повреждения и его свойства. Таким образом, недеструктивными методами SHM являются разные способы измерения параметров системы и их последующей обработки. Способы измерения могут быть разделены по типу задачи:

- Статическая задача – измерение деформации или сдвигов разных частей конструкции под нагрузкой [4], [5].
- Динамическая задача – измерение собственных частот и/или нахождение форм колебаний при этих частотах. [2], [3], [6].

В этой работе применяется первый способ.

В качестве испытательного образца рассматривается консольная балка со скрытым дефектом, имеющим вид расслоения – распространенного дефекта слоистых композитов. Свободный

конец балки нагружается известной силой F. Снимаются показания тензодатчиков, расположенных на верхней грани балки. Тензодатчики показывают перемещения точек балки вдоль оси, возникающие вследствие деформации. Наличие, величина и местонахождение повреждения влияют на полученные данные, а значит возможно создание алгоритма, решающего обратную задачу. Вместо реальных экспериментов с балками были созданы их конечно-элементные модели в программном пакете Wolfram Mathematica.

Балки используются в качестве образцов в большинстве рассмотренных автором статей, поскольку они просты в изучении, и в то же время успешная проверка алгоритмов на их примере открывает путь к пониманию поведения более сложных конструкций [3].

Рассматривается *задача классификации*, то есть на выходе алгоритм выдает распределение вероятностей местонахождения/величины дефекта из набора дискретных значений. В качестве метода решения задачи было выбрано *обучение с учителем* – один из методов машинного обучения, при котором алгоритм обучается по достаточно большому набору пар «стимул-реакция», которые называются *обучающей выборкой*. Обучающая выборка программно генерируется для каждой задачи из 10 000 экспериментов со случайно выбранными параметрами повреждения. В качестве обучающегося на этих данных алгоритма используется *искусственная нейронная сеть* (ИНС).

ИНС являются одним из классов систем *анализа больших данных* – компьютерной дисциплины, фокусирующейся на обработке больших объемов "сырых" данных, то есть непригодных для человеческого восприятия в исходном виде. В этих данных требуется обнаружить с помощью алгоритмов некие закономерности, или *паттерны*, на основе которых можно, например, научиться делать предсказания для новых данных. В данном случае паттерном служит взаимосвязь между параметрами системы и местонахождением и величиной дефекта. Примеры других классов систем:

- Деревья решений – классификация при помощи деревьев, узлы которых содержат либо значения целевой функции ("листья"), либо атрибуты, по которым различаются случаи. Для классификации нового случая надо спуститься по дереву до одного из листьев.
- Эволюционное программирование – процесс построения программ, организованный как эволюция в мире программ: система вносит в свои дочерние программы модификации и отбирает те из них, которые являются наиболее точными.
- Кластерный анализ – процедура упорядочивания многомерных объектов в сравнительно однородные группы. Относится к задачам обучения без учителя.

Использование ИНС для идентификации дефектов описано в [4], [5], [6]. В работе [4] ИНС применяются для нахождения местоположения и величины повреждения в Т-образном полимерном композите. Предсказания двух независимо обученных ИНС обрабатываются затем третьей, глобальной ИНС. В [5] ставится задача идентификации дефектов в консольной балке, однако дефект моделируется домножением матрицы жесткости конечных элементов на заданный коэффициент. Проверяется способность ИНС к обучению отдельно на перемещениях и деформациях модели. Наконец, в [6] ставится динамическая задача для балки со свободными концами; дефект моделируется домножением модулей Юнга отдельных участков балки на заданные коэффициенты. Используется система из целого дерева нейронных сетей, на вход системы подаются формы колебаний и собственные частоты балки. Стоит отметить, что архитектура ИНС определяется множеством параметров, которые подбираются отдельно для каждой задачи, и зависят во многом от особенностей обучающей выборки. Подробнее об этом в разделе 4.

Для написания программ в этой работе использовались языки Wolfram с пакетом NDSolve‘FEM‘ и Python 3.6.4 с open-source библиотеками NumPy, MatPlotLib, TensorFlow, Keras.

2 Постановка задачи

Рассматривается балка Эйлера-Бернулли. То есть предполагается, что выполняются следующие гипотезы [3]:

- Материал изотропный, однородный, линейно-упругий.
- Выполняется гипотеза Эйлера-Бернулли (гипотеза плоских сечений), т. е. не учитываем сдвиговые деформации.
- Левый конец жестко зафиксирован (граничное условие Дирихле).
- Правый конец нагружен известной силой F (граничное условие Неймана).

Дефект моделируется полостью в виде эллипса (Рис. 1). В качестве входных данных алгоритма используются показания 11 тензодатчиков, расположенных на верхней грани балки. Будем считать, что тензодатчики показывают относительное удлинение, поэтому их показания можно моделировать, просто получая главную компоненту ε_{11} тензора деформаций. Модуль Юнга взят равным $E = 12.7 \cdot 10^6$Па, коэффициент Пуассона $\nu = 0.33$, длина и высота балки $L = 2$м, $h = 0.2$м. Координаты центра дефекта выбираются произвольно, так чтобы эллипс не выходил за границы балки. Малая полуось эллипса $a = 0.02$м, большая полуось $b \in [0.04\text{м}, 0.16\text{м}]$, варьированием величины b моделируется размер дефекта. Сила F принимается равной либо 2кг, либо 4кг, с целью выяснить, при каком значении силы получаются более точные результаты.

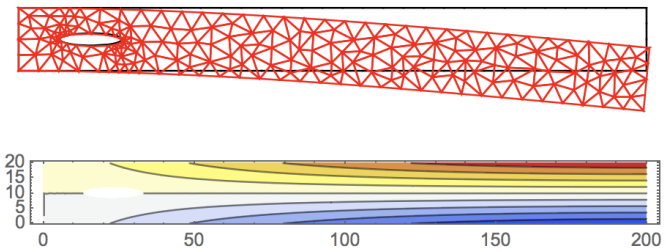


Рис. 1: Пример балки с повреждением: модель балки и диаграмма значений ε_{11}.

Разделим балку на 10 равных участков и обозначим i-ый участок $elem_i$:

$$elem_i = \{x : \ x \in [i \cdot 0.2\text{м}, (i+1) \cdot 0.2\text{м}]\}, \ i = 0 \ldots 9 \tag{1}$$

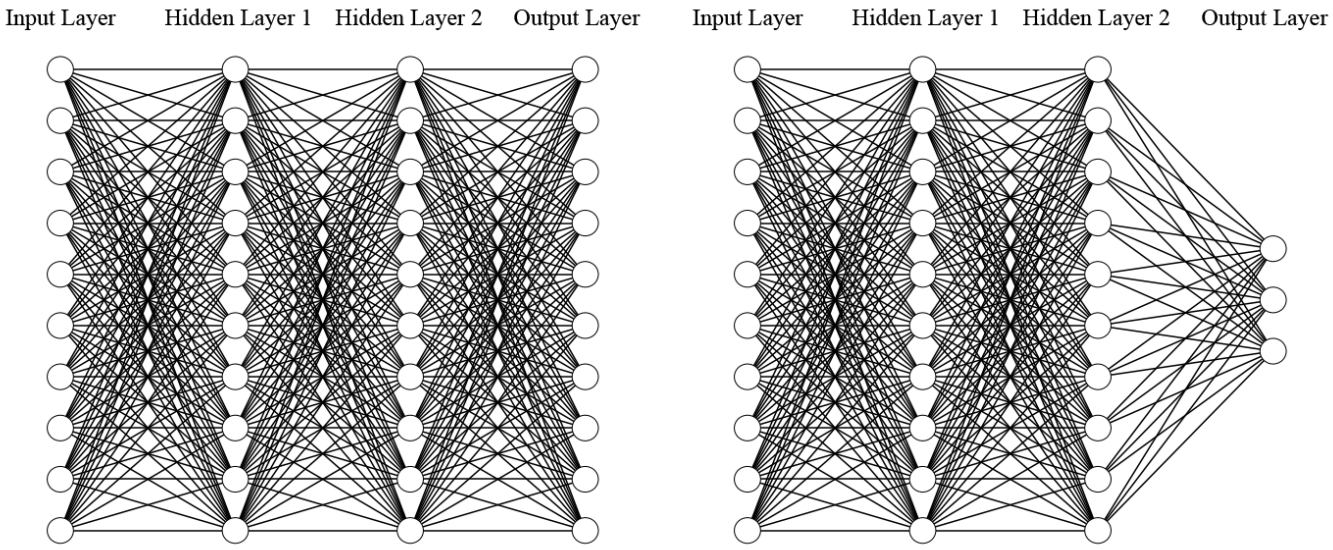
Аналогично разделим множество возможных значений b на 3 равных участка:

$$ex_j = \{b : \ b \in [j \cdot 0.04\text{м} + 0.04\text{м}, (j+1) \cdot 0.04\text{м} + 0.04\text{м}]\}, \ j = 0 \ldots 2 \tag{2}$$

Задача ставится таким образом: по показаниям тензодатчиков требуется определить i и j, такие что для координаты x повреждения и его величины ex верно, что $x \in elem_i$ и $ex \in ex_j$. Другими словами, надо указать, в каком участке балки и каком отрезке из множества возможных размеров повреждения лежит данное повреждение. Эта задача называется задачей классификации.

3 Принцип работы искусственных нейронных сетей

Искусственная нейронная сеть – это математическая модель, построенная по принципу функционирования сетей нервных клеток живого организма. ИНС может быть представлена в виде взвешенного графа, вершины которого называются *нейронами*, а веса ребер называются *весами нейронов*. На Рис. 2 представлена топология нейронных сетей, используемая в этой работе. Нейроны группируются в *слои* (вертикальные столбцы на Рис. 2), слои делятся на три вида: *входной, скрытый, выходной*. В этой работе используется по два скрытых слоя в каждой нейронной сети. В ходе работы нейронной сети сигнал поступает на входной слой и проходит слева направо через все слои, преобразуясь определенным образом. Такие нейронные сети называются сетями *прямого распространения* или *перцептронами*.



(a) 10 нейронов в выходном слое

(b) 3 нейрона в выходном слое

Рис. 2: Топология нейронных сетей, используемых в этой работе. Различие только в количестве нейронов в выходном слое (Output Layer).

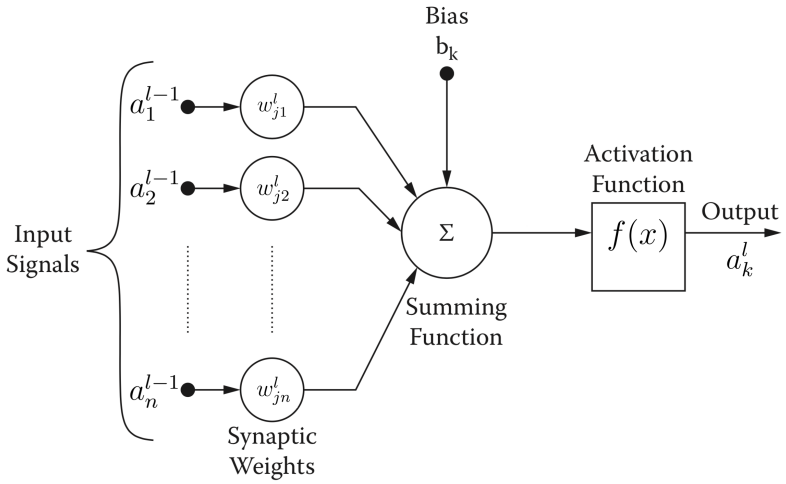
Нейронные сети – это алгоритмы, работающие по принципу *обучения с учителем*. Это значит, что перед тем как можно будет использовать алгоритм на практике, надо провести *обучение*. Процесс обучения является итерационным процессом и состоит в корректировке весов нейронов. Для этого используется *обучающая выборка* – набор входных данных и соответствующий им набор заранее известных выходных данных. Этот набор также иногда называют набором пар «стимул-реакция».

Для описания принципа работы ИНН необходимо дать описание нейрона, процесса *прямого распространения сигнала* и *обратного распространения ошибки*.

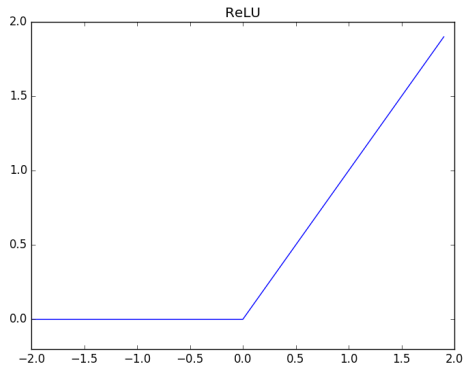
3.1 Устройство нейрона

Нейрон представляет собой линейную комбинацию поступающих в него сигналов $a_1^{l-1} \dots a_n^{l-1}$, взятых с коэффициентами $w_{1k}^l \dots w_{nk}^l$, называемых *весами* нейрона. Здесь w_{jk}^l обозначен вес, стоящий на ребре, соединяющем j-й нейрон слоя $(l-1)$ и k-й нейрон слоя l. К линейной комбинации иногда добавляют *смещение* b_k, но в этой работе смещение не используется. От линейной комбинации затем берется *функция активации* $f(x)$. Результат её вычисления и является выходным сигналом нейрона. Описанная схема работы приведена на Рис. 3а. Математический эквивалент этой схемы можно записать как (3).

$$a_k^l = f(\sum_{j=1}^{n} a_j^{l-1} w_{jk}^l + b_k) \tag{3}$$



(a) схема работы нейрона



(b) график функции ReLU

Рис. 3: Модель нейрона.

В представленной работе в качестве функции активации на скрытых слоях используется ReLU (Rectified Linear Unit):

$$a_k^l = relu(z_k^l) = max(0, z_k^l) \tag{4}$$

Её график изображен на Рис. 3b. Эта функция была выбрана, поскольку она значительно ускоряет сходимость градиентного спуска во время обратного распространения ошибки по сравнению с функциями $sigmoid(x)$ и $tanh(x)$ [7].

В качестве функции активации на выходном слое используется softmax:

$$a_k^L = softmax(z_k^L) = \frac{e^{z_k^L}}{\sum_c e^{z_c^L}} \tag{5}$$

Это функция, преобразующая вектор z размерности K в вектор a той же размерности, где каждая координата a_k полученного вектора представлена вещественным числом в отрезке $[0, 1]$ и сумма координат равна 1. Таким образом мы получаем на выходе распределение вероятностей. Здесь и далее буквой L обозначается индекс последнего – выходного – слоя.

3.2 Прямое распространение сигнала

Прямое распространение сигнала – это процесс преобразования входных данных в конечный результат работы нейросети. Прямое распространение используется как во время обучения – чтобы найти вектор ошибки – так и во время работы на новых данных, чтобы получить искомый результат.

Посмотрим, как получается аргумент функции активации первого слоя:

$$z_k^1 = \sum_j a_j^0 w_{jk}^1 \tag{6}$$

Здесь a_j^0 – тензор входных данных. Аналогично для второго слоя:

$$\begin{aligned} z_k^2 &= \sum_j f^1(z_j^1) w_{jk}^2 \\ z_k^2 &= \sum_j a_j^1 w_{jk}^2 \end{aligned} \tag{7}$$

Это удобно представить в матричном виде, заменив вектор входных данных a_j^0 на матрицу входных данных A_{ij}^0. Таким образом вместо одного набора входных данных за один проход

обрабатывается N наборов. Число N соответствует числу строк в матрице A^0 и называется размером батча (batch size):

$$\begin{aligned} Z^1 &= A^0 W^1 \\ Z^1_{ik} &= \sum_j A^0_{ij} W^1_{jk} \end{aligned} \tag{8}$$

Весь процесс прямого распространения сигнала можно записать через матричные операции:

$$\begin{aligned} Z^1 &= A^0 W^1 \\ A^1 &= f^1(Z^1) \\ Z^2 &= A^1 W^2 \\ A^2 &= f^2(Z^2) \\ Z^3 &= A^2 W^3 \\ A^3 &= f^3(Z^3) \end{aligned} \tag{9}$$

Или:

$$A^3 = f^3(f^2(f^1(A^0 W^1) W^2) W^3) \tag{10}$$

3.3 Обратное распространение ошибки

Для корректировки весов нейронов необходимо знать вектор ошибки на выходном слое, то есть разницу между желаемым результатом и полученным. Напомним, что результат – это некое дискретное распределение вероятностей, т. к. в качестве активационной функции на выходном слое используется softmax (5).

В этой работе в качестве функции ошибки используется перекрестная энтропия:

$$E(t_k, a^L_k) = -\sum_k t_k \ln a^L_k = -\sum_k t_k (z^L_k - \ln \sum_c e^{z^L_c}) \tag{11}$$

Здесь t_k – это желаемое распределение, которое имеет вид $\{0, \ldots, 0, 1, 0, \ldots, 0\}$, а a^L_k – полученное распределение, то есть сигнал на выходе последнего слоя L. Перекрестная энтропия здесь используется в качестве меры различия между двумя распределениями: чем больше различие, тем больше E.

Для корректировки весов нейронов к каждому весу должна быть прибавлена поправка

$$\Delta w^l_{jk} = -\eta \frac{\partial E}{\partial a^l_l} \tag{12}$$

Сама процедура называется *градиентным спуском*, а η *шагом* градиентного спуска. Рассмотрим как вычисляется эта производная для последнего слоя и для скрытых слоев.

На последнем (выходном) слое

$$\frac{\partial E}{\partial w^L_{jk}} = \frac{\partial E}{\partial z^L_k} \frac{\partial z^L_k}{\partial w^L_{jk}} \tag{13}$$

Посчитаем первый множитель, продифференцировав (11):

$$\begin{aligned} \frac{\partial E}{\partial z^L_k} &= -\sum_d t_d (I_{d=k} - \frac{1}{\sum_c e^{z^L_c}} e^{z^L_k}) \\ &= -\sum_d t_d (I_{d=k} - a^L_k) \\ &= \sum_d t_d a^L_k - \sum_d t_d I_{d=k} \\ &= a^L_k \sum_d t_d - t_k \\ &= a^L_k - t_k \end{aligned} \tag{14}$$

Здесь $I_{d=k}$ – единичный тензор. Второй множитель считается тривиально:

$$\frac{\partial z_k^L}{\partial w_{jk}^L} = a_j^{L-1} \tag{15}$$

Тогда, обозначив

$$\delta_k^L = \frac{\partial E}{\partial z_k^L} = a_k^L - t_k, \tag{16}$$

можем записать результат в виде:

$$\frac{\partial E}{\partial w_{jk}^L} = \delta_k^L a_j^{L-1} \tag{17}$$

Проделаем аналогичные вычисления для скрытого слоя $l-1$.

$$\frac{\partial E}{\partial w_{ij}^{l-1}} = \frac{\partial E}{\partial a_j^{l-1}} \frac{\partial a_j^{l-1}}{\partial z_j^{l-1}} \frac{\partial z_j^{l-1}}{\partial w_{ij}^{l-1}} \tag{18}$$

$$\begin{aligned} \frac{\partial E}{\partial a_j^{l-1}} &= \sum_k \frac{\partial E}{\partial z_k^l} \frac{\partial z_k^l}{\partial a_j^{l-1}} \\ &= \sum_k \delta_k^l w_{jk}^l \end{aligned} \tag{19}$$

$$\frac{\partial a_j^{l-1}}{\partial z_j^{l-1}} = f'(z_j^{l-1}) \tag{20}$$

$$\frac{\partial z_j^{l-1}}{\partial w_{ij}^{l-1}} = a_i^{l-2} \tag{21}$$

Объединяя всё вместе, получаем:

$$\frac{\partial E}{\partial w_{ij}^{l-1}} = a_i^{l-2} f'(z_j^{l-1}) \sum_k \delta_k^l w_{jk}^l \tag{22}$$

Обозначим

$$\delta_j^{l-1} = \frac{\partial E}{\partial a_j^{l-1}} \frac{\partial a_j^{l-1}}{\partial z_j^{l-1}} = f'(z_j^{l-1}) \sum_k \delta_k^l w_{jk}^l \tag{23}$$

и перепишем выражение (22):

$$\frac{\partial E}{\partial w_{ij}^{l-1}} = \delta_j^{l-1} a_i^{l-2} \tag{24}$$

Таким образом, мы научились вычислять поправку для узлов последнего уровня и выражать поправку для узла более низкого уровня через поправки более высокого. Именно из-за этой особенности вычисления поправок алгоритм называется алгоритмом обратного распространения ошибки (backpropagation).

4 Описание решения

4.1 Получение обучающих выборок

Все выборки состоят из 10 000 моделей балки со случайно варьируемыми параметрами дефекта.

В первой выборке set_0 варьируются координаты (x, y) центра повреждения, а остальные параметры повреждения постоянны. В базу данных для каждой модели сохраняется одна строка, состоящая из координат повреждения и показаний датчиков (Рис. 4). Показания 0-го датчика не используются при обучении, потому что по постановке задачи они всегда нулевые. Координата y также игнорируется, а x преобразуется в индекс i участка $elem_i$ целочисленным делением на 20. Индекс i называется меткой (label) и используется в качестве правильного ответа при обучении. Были сформированы два варианта этой выборки для случая $F = 2$кг и $F = 4$кг. Эти выборки используются для задачи предсказания местонахождения повреждения при фиксированной величине.

output1_2kg

exp_number	crackX	crackY	sensor0	sensor1	sensor2	sensor3	sensor4	sensor5	sensor6	sensor7	sensor8	sensor9	sensor10
0	54.2316	9.56525	0.000000000000000000693889	0.178669	0.339838	0.484819	0.603339	0.707279	0.792318	0.858462	0.905704	0.934043	0.944035
1	125.947	13.1096	0.	0.178667	0.339277	0.481023	0.603845	0.707771	0.799335	0.868086	0.917819	0.946136	0.956128
2	55.5801	10.4106	0.	0.178665	0.339822	0.486196	0.606208	0.710181	0.795219	0.861363	0.908605	0.936944	0.946936
3	21.0162	12.6686	0.	0.192806	0.360446	0.503292	0.626105	0.730048	0.815087	0.881231	0.928472	0.956812	0.966803
4	136.74	8.34734	0.	0.178662	0.339295	0.48101	0.603855	0.707786	0.793153	0.863998	0.905384	0.933644	0.943636
5	45.2895	12.8341	0.	0.178662	0.351345	0.498852	0.624928	0.728835	0.813874	0.880018	0.92726	0.955599	0.96559

Рис. 4: Пример содержания базы данных.

Была также сформирована выборка set_1, отличающаяся от set_0 тем, что дополнительно варьировался размер повреждения. Соответственно, в базу данных добавился столбец crack_extent. Эта выборка используется для задачи предсказания местонахождения повреждения при неизвестном размере повреждения.

Наконец, были сформированы выборки $set_{2i}, i = 0 \dots 9$. Они аналогичны set_1, за исключением того, что координаты x дефектов выбирались таким образом, что $x \in elem_i$. При обучении по этой выборке значения столбца преобразовывались в метку j размера дефекта (см. раздел 2). Эти выборки используются для задачи предсказания размера дефекта при известном местонахождении (см. раздел 4).

4.2 Параметры нейронных сетей

Во всех нейронных сетях использовалась архитектура с двумя скрытыми слоями, по 10 нейронов в каждом, с функциями активации ReLU на скрытых слоях и softmax на выходном слое (см. раздел 3).

Особенностью решения является то, что вместо обычного градиентного спуска используется алгоритм AdaGrad (Adaptive Gradient Descent) [8]. Улучшение состоит в том, что шаг η зависит от конкретного параметра (входа нейрона в данном случае) и временного шага. То есть, если обычный градиентный спуск работает по принципу

$$\Delta w_{jk}^l = -\eta \frac{\partial E}{\partial a_l^l} \ , \tag{25}$$

то AdaGrad модифицирует шаг градиентного спуска на каждом временном шаге для каждого w_{jk}^l по принципу:

$$\Delta w_{jk}^l = -\frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \frac{\partial E}{\partial a_l^l} \ , \tag{26}$$

где $G_{t,ii}$ – диагональная матрица, где каждый диагональный элемент g_{ii} является суммой квадратов градиентов, вычисленных для w_{jk}^l за все моменты времени до момента t. Здесь ε это

малое слагаемое, введенное для избежания деления на 0. Таким образом, AdaGrad подстраивает шаг таким образом, чтобы со временем замедлять слишком быстро меняющиеся веса. AdaGrad показал наилучшие результаты на полученных выборках данных.

Процесс обучения продолжается до тех пор, пока не начнется *переобучение*, либо пока не перестанет расти точность. Переобучение – это явление, при котором нейронная сеть излишне приспосабливается к обучающим данным и потери (loss) (11) начинают расти на тестовой выборке. При использовании AdaGrad обучение было прекращено по причине того, что перестала расти точность (см. раздел 4).

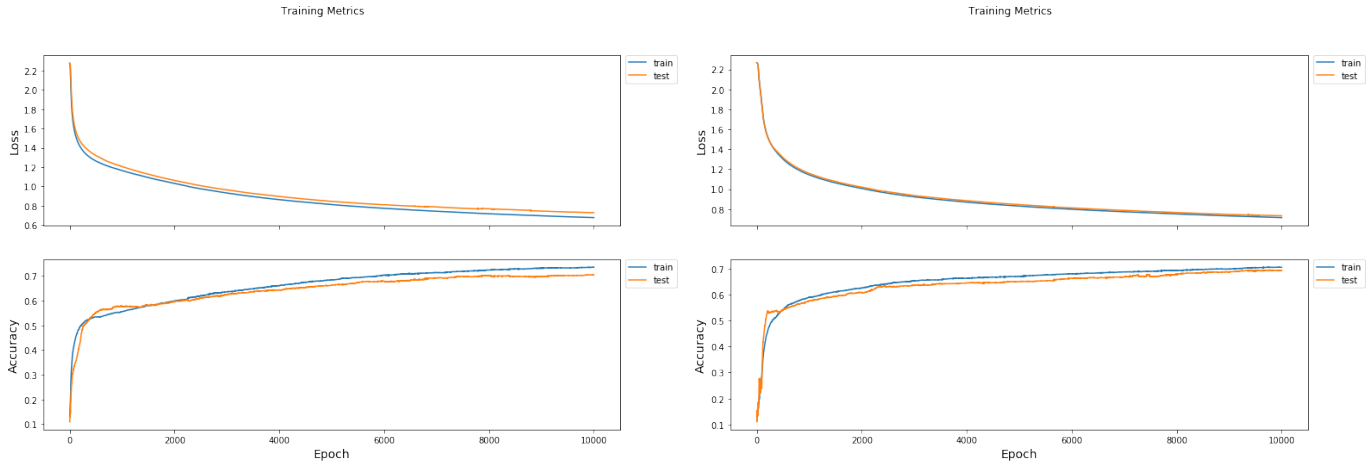
5 Результаты

Ниже приведены результаты обучения нейросетей, имеющих сходную архитектуру, описанную в разделе 4, но обученных на разных выборках.

5.1 Прогнозирование местонахождения повреждения при его фиксированной величине

Выборка состоит из 10 000 моделей балки, в каждой из которых случайно выбраны координаты (x, y) центра повреждения, а остальные параметры повреждения постоянны. Выборка поделена на обучающую и тестовую в отношении 4 : 1. То есть, на каждой эпохе обучения 8000 строк данных используются для подстройки весов, а 2000 только для проверки. Размер батча (batch) взят равным 64.

Всего для каждого варианта нагрузки (2кг и 4кг) было проведено по 10 000 эпох обучения (Рис. 5). На каждом графике представлена зависимость потерь (loss) или точности (accuracy) от количества эпох. Видно, что линии, соответствующие тестовой выборке, немного отстают от обучающей, но при этом разрыв значительно не увеличивается. Это говорит о том, что *переобучения* не происходит. То есть, не происходит излишнего приспособления нейросети к обучающим данным, из-за которого она перестает делать правильные предсказания на тестовых.



(a) нагрузка 2кг

(b) нагрузка 4кг

Рис. 5: Графики потерь (loss) и точности (accuracy) на обучающей и тестовой выборке для нейронных сетей, обученных на моделях с разной нагрузкой.

Удалось добиться следующих результатов на 10 000-й эпохе:

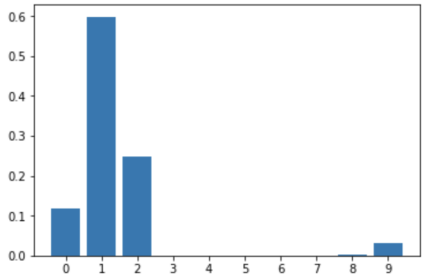
Нагрузка	Выборка	Потери (Loss)	Точность (Accuracy)
2кг	Train	0.678	73.550%
2кг	Test	0.729	70.376%
4кг	Train	0.715	70.450%
4кг	Test	0.733	69.200%

Видно, что при обучении с нагрузкой 2кг удалось получить небольшой выигрыш в точности: всего 1.176%. Таким образом, при выборе одного поврежденного участка из 10, нейросеть дает верный результат примерно в 70% случаев.

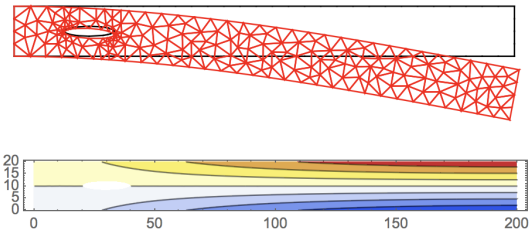
Интересно посмотреть, какие предсказания сделает нейросеть на вручную выбранных данных, а не сгенерированных случайно. На Рис. 6 приведен такой пример. Тензор входных данных состоит из показаний датчиков; тензор предсказаний содержит выходные значения сигналов на последнем слое нейросети (Рис. 6a). Тензор предсказаний затем преобразуется в распределение вероятностей функцией softmax (5) и отображается на диаграмме. По диаграмме видно,

что наиболее вероятно центр повреждения лежит в участке под индексом 1 – это соответствует истине.

```
input tensor: tf.Tensor(
[[0.18082097 0.34134886 0.48163694 0.6044725  0.708416   0.7934547
  0.85959816 0.9068401  0.9351795  0.945171  ]], shape=(1, 10), dtype=float32)
prediction: tf.Tensor(
[[ -5.1013575  -3.4690087  -4.351266  -10.599114  -13.607382  -10.235837
  -10.893956  -14.897898   -8.612098   -6.408027 ]], shape=(1, 10), dtype=float32)
```



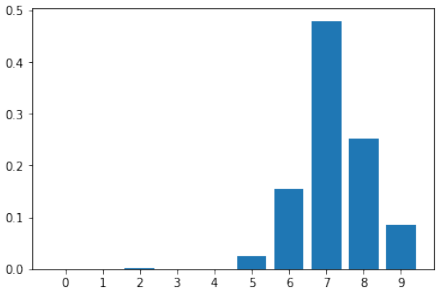
(a) предсказание



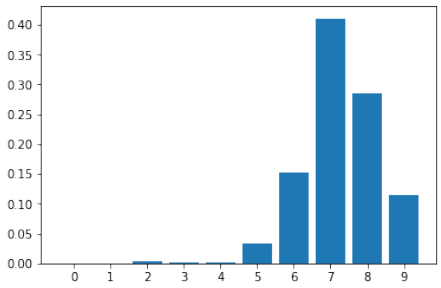
(b) соответствующая модель

Рис. 6: Тест на модели с повреждением в координатах $(30, 10)$.

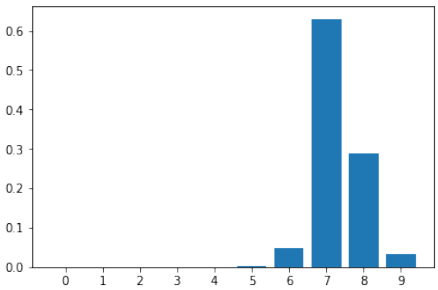
В этом случае предсказание весьма уверенное – лидирующая вероятность более чем на 0.3 превышает следующую за ней. Рассмотрим случай менее уверенного предсказания – для повреждения в координатах $(150, 10)$ (Рис. 7b). Выбором разных y-координат повреждения (Рис. 7a, 7c) убеждаемся, что точность предсказаний увеличивается при удалении повреждения от срединной линии балки. Об этом говорит значение вероятности нахождения повреждения в участке под индексом 7, обозначенной как $P(elem_7)$.



(a) $y = 6 : P(elem_7) \approx 0.5$



(b) $y = 10 : P(elem_7) \approx 0.4$



(c) $y = 14 : P(elem_7) \approx 0.6$

Рис. 7: Тест на моделях с повреждениями в $x = 150$ и разными y.

Интересно также рассмотреть случай, когда центр повреждения лежит не посередине какого-либо участка, а ровно на границе между двумя. Например, возьмем центр в $x = 80$, то есть на границе между 3 и 4 участками (Рис. 8). По распределению вероятностей видно, что столбцы, отвечающие 3 и 4 участкам, имеют примерно равные высоты, отличающиеся не более чем на 0.05. Это наблюдение можно использовать на практике: при близких соседних вероятностях, когда прочие вероятности малы, скорее всего повреждение находится где-то посередине.

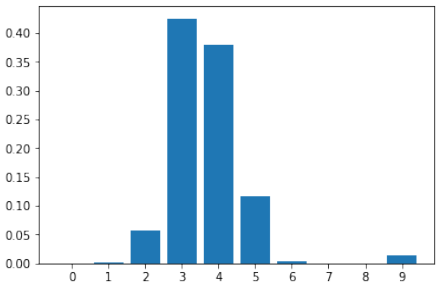
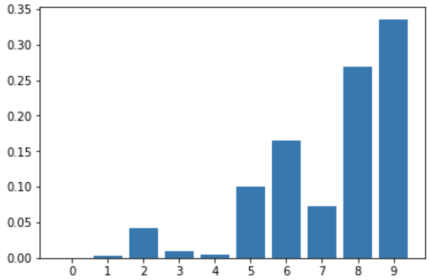


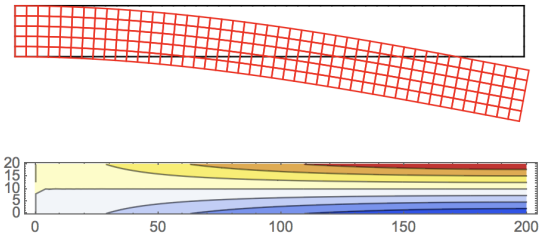
Рис. 8: Тест на модели с повреждением в координатах $(80, 10)$.

Наконец, посмотрим на предсказание нейросети для модели, вообще не имеющей повреждения (Рис. 9). Это предсказание весьма логично. Ведь, напомним, нейросеть априори "полагает", что в балке есть повреждение. Стало быть, не находя признаков этого повреждения, она смещает свой прогноз в ту часть балки, где повреждение наименее всего влияет на показания датчиков, то есть в правый конец.

```
input tensor: tf.Tensor(
[[0.17867227 0.3393038  0.481035   0.60387075 0.70780665 0.7928471
  0.85898775 0.90623295 0.9345781  0.9446501 ]], shape=(1, 10), dtype=float32)
prediction: tf.Tensor(
[[-13.074495   -9.409701   -6.7724066  -8.272366   -8.93003    -5.8753896
   -5.387417   -6.204199   -4.894806   -4.672768 ]], shape=(1, 10), dtype=float32)
```



(a) предсказание



(b) соответствующая модель

Рис. 9: Тест на модели без повреждения.

Несмотря на логичность, нейросеть, очевидно, плохой инструмент для проверки факта наличия дефекта. Вместо этого надо просто перед применением нейросети проверять, не совпадает ли тензор входных данных с тензором из первой строчки Рис. 9a в пределах погрешности датчиков. Если совпадает, значит применение нейросети вовсе не требуется – повреждение отсутствует.

5.2 Прогнозирование местонахождения и величины повреждения

Для решения усложненной задачи поиска двух параметров повреждения использовалась следующая система.

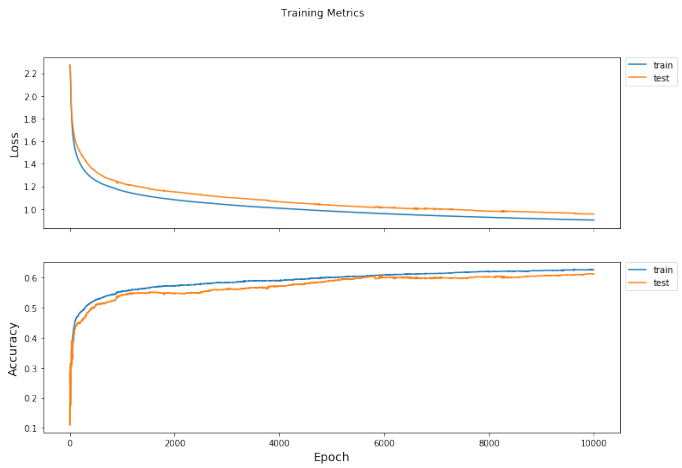


Рис. 10: Графики потерь (loss) и точности (accuracy) на обучающей и тестовой выборке для nn_1.

Сначала нейросеть находит участок с повреждением, аналогично тому как это делалось в разделе 5.1. Для этого опять же использовалась выборка из 10 000 моделей, но помимо координат (x, y) добавился третий, также случайно варьируемый параметр ex (см. раздел 2). Обозначим эту выборку set_1, а соответствующую нейросеть – nn_1. Для каждого участка i была обучена отдельная нейросеть nn_{2i} по выборке set_{2i} из 10 000 моделей, в которой $(x, y) \in elem_i$, $ex \in D_{ex}$ (D_{ex} – вся область определения ex). На выходе эти нейросети дают только оценку величины повреждения.

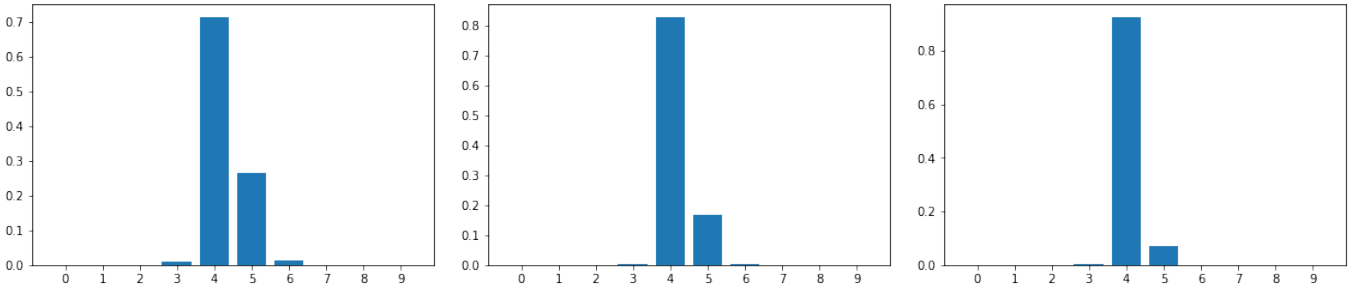
Таким образом, обозначив тензор входных данных a_k^0, процесс получения результата можно описать двумя шагами:

1. Подаем тензор a_k^0 на вход нейросети $nn_1 \Rightarrow$ получаем индекс i участка, содержащего центр повреждения.
2. Подаем тензор a_k^0 на вход нейросети $nn_{2i} \Rightarrow$ получаем оценку величины повреждения ex.

На графиках (Рис. 10) показан процесс обучения нейросети nn_1. Результаты, полученные на 10 000-й эпохе, показаны в таблице:

Нагрузка	Выборка	Потери (Loss)	Точность (Accuracy)
2кг	Train	0.901	62.513%
2кг	Test	0.955	61.150%

За счет дополнительного случайного параметра ex точность упала примерно на 9% по сравнению с аналогичными результатами в разделе 5.1.



(a) $ex = 6 : P(elem_4) \approx 0.7$ (b) $ex = 10 : P(elem_4) \approx 0.8$ (c) $ex = 14 : P(elem_4) \approx 0.85$

Рис. 11: Тест нейросети nn_1 на моделях с повреждениями в $x = 90,\ y = 14$ и разными ex.

Будем обозначать повреждения тройками $\{x, y, ex\}$. Проверим, например, правильность предсказаний для трех моделей с параметрами повреждений $\{90, 14, 6\}$, $\{90, 14, 10\}$, $\{90, 14, 14\}$. Нейросеть для всех трех моделей дала правильный результат (Рис. 11). Что характерно, точность предсказаний увеличивается при увеличении размера повреждения.

Поскольку результат – участок балки под индексом 4, для предсказания величины повреждения выбираем нейросеть nn_{24}. Графики обучения приведены на Рис. 12. Архитектура этой нейросети отличается от предыдущих только числом нейронов в выходном слое – 3 вместо 10 – и размером батча – 32 вместо 64. По Рис. 13 видно, что нейросеть дала верные результаты на трех выбранных моделях.

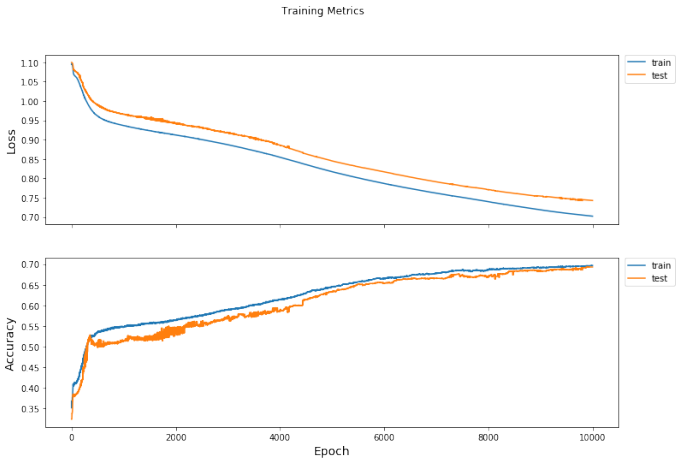
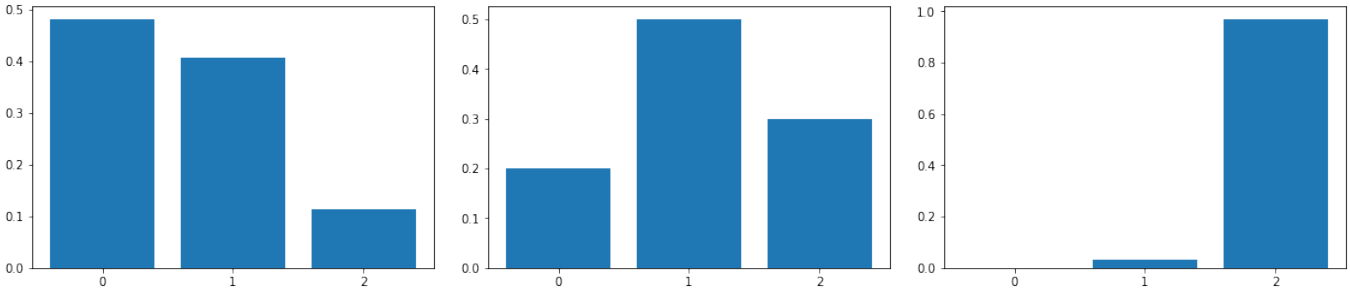


Рис. 12: Графики потерь (loss) и точности (accuracy) на обучающей и тестовой выборке для nn_{24}.



(a) $ex = 6$ (b) $ex = 10$ (c) $ex = 14$

Рис. 13: Тест нейросети nn_{24} на моделях с повреждениями в $x = 90,\ y = 14$ и разными ex.

Результаты, полученные на 10 000-й эпохе:

Нагрузка	Выборка	Потери (Loss)	Точность (Accuracy)
2кг	Train	0.702	69.675%
2кг	Test	0.743	69.300%

6 Заключение

В работе предложена общая методология идентификации дефектов с помощью нейронных сетей, а также реализован конкретный алгоритм. Алгоритм применен к статической задаче с консольной балкой с дефектом. Были подобраны параметры, на которых алгоритм дал наилучшие результаты, а также протестирована точность на моделях с разными величинами нагрузки. Можно считать, что опыт по применению нейронных сетей к задаче идентификации дефектов показал положительные результаты.

7 Список литературы

[1] Jean-Jacques Sinou, A Review of Damage Detection and Health Monitoring of Mechanical Systems from Changes in the Measurement of Linear and Non-linear Vibrations, 2009.

[2] Dawid Cekus, Mateusz Miara, Izabela Zamorska, Damage Identification of a Beam With a Variable Cross-Sectional Area, 2016.

[3] Maged Mostafa, Mohammad Tawfik, Crack Detection Using Mixed Axial and Bending Natural Frequencies in Metallic Euler-Bernoulli Beam, 2016.

[4] S. John, A. Kesavan, and I. Herszberg, The Use of Artificial Neural Networks in Damage Detection and Assessment in Polymeric Composite Structures, 2010.

[5] Ismoyo Haryanto, Joga Dharma Setiawan, Agus Budiyono, Structural Damage Detection Using Randomized Trained Neural Networks, 2007.

[6] Norhisham Bakhary, Hong Hao, Andrew John Deeks, Structure Damage Detection Using Neural Network with Multi-Stage Substructuring, 2010.

[7] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks.

[8] John Duchi, Elad Hazan, Yoram Singer, Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, 2011.