

Федеральное государственное автономное
образовательное
Учреждение высшего профессионального образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет МИЭМ
Департамент прикладной математики

Моя первая нейронная сеть. Распознавание лиц.

Исследовательская работа по дисциплине Профорientационный семинар
«Введение в специальность»

Выполнил студент группы БПМ234:
Карпов Аркадий Дмитриевич

Научный руководитель:
Попов Виктор Юрьевич

Москва 2024

Содержание

1 Введение	2
2 Теоретическая часть	4
3 Практическая часть	7
4 Вывод	17
5 Приложение	18
6 Список источников и мои материалы	19

1 Введение

Одной из самых перспективных сфер применения искусственного интеллекта является обработка изображений с помощью нейронных сетей. С помощью их способности автоматически учиться, можно решать сложные задачи в различных сферах.

Нейронные сети (НС) — это совокупность алгоритмов машинного обучения (machine learning), напоминающая собой структуру биологических нейронов в мозгу. Строение нейронных сетей представляет из себя несколько слоев искусственных нейронов соединённых между собой весами.

Машинное обучение (Machine Learning) - это технология обучения компьютера с помощью входных данных, что позволяет добиться высокой эффективности в решение большого количества сложных задач.

Глубокое обучение (Deep learning) – это способ решения различных сложных задач в области ML с помощью искусственных нейронных сетей. Например, задач прогнозирования, классификации, сегментации и обнаружения объектов.

Обработка изображений - одна из самых перспективных областей применения машинного обучения в мире. Нейросети, связанные с обработкой изображений, обучающиеся на больших данных, выявляют закономерности, что позволяет решать различные задачи, такие как распознавание лиц и обнаружение объектов. Также они получили свое применение в сферах здравоохранения, безопасности, маркетинга и многих других.

Объект исследования: Нейросеть, которая будет решать задачу распознавания лиц по изображению.

Предметом исследования: Написание модели способной делать верификацию человека.

Цель работы: Изучить строение нейронных сетей и написать модель, которая будет решать задачу распознавания лиц.

Задачи:

- Изучение строения и работы нейронных сетей распознавания лиц по изображениям.
- Подготовка данных для обучения нейросети.
- Разработка и обучение сверточной сямской нейронной сети для распознавания лиц по изображениям.
- Тесты на своих фотографиях и фотографиях других людей.
- Отчет будет состоять из теоретических основ и описания разработки нейронной сети.

2 Теоретическая часть

Сиамская сверточная нейронная сеть (Siamese Convolution Neural Network) - представляет собой нейронную сеть, которая содержит в своей архитектуре две и более одинаковые свёрточные подсети, которые используются для задач сопоставления, они обучаются одновременно на больших данных, что позволяет им выявлять закономерности в изображениях.

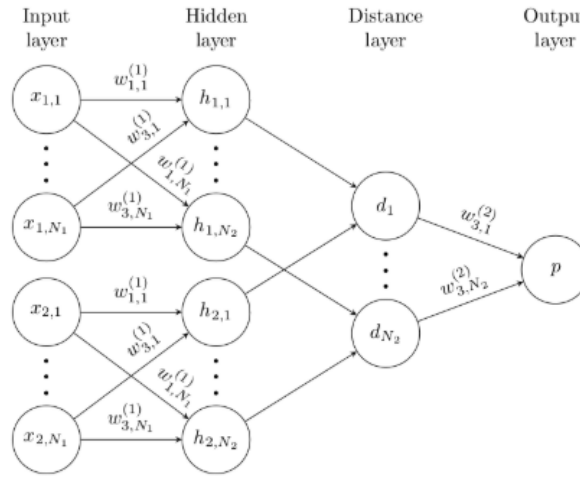


Рис. 1: Пример двухслойной сиамской НС

Backpropagation – широко используемый метод обучения нейронных сетей. Его предназначение — это настройка весов и минимизация функции потерь. Также его называют «алгоритм обратного распространения ошибки».

Алгоритм работы backpropagation:

1. Инициализация весов: в самом начале обучения веса нейронной сети задаются случайными значениями.

2. Прямой проход: на вход нейронной сети подаются обучающие данные (обучающая выборка). Они проходят через все слои. В конце получается результат, который сравнивается с фактическим значением (меткой) из обучающей выборки.

3. Вычисление ошибки: на основе сравнения результата и значения метки вычисляется величина ошибки. Она будет использоваться для корректировки весов между слоями в будущем.

4. Обратный проход: с помощью вычисления градиента отношения ошибки к весу на каждом слое происходит корректировка весов.

5. Обновление весов: после корректировки весов на каждом слое, алгоритм начинается заново и повторяется до достижения по не будет достигнут определенный уровень точности.

Оптимизаторы-алгоритмы, которые, уменьшая функцию потерь, позволяют ускорить настройку весов во время обучения НС. Популярные оптимизаторы: Gradient Descent, Adam, Momentum, RMSprop и Adagrad.

Для обучения нейросети нужно подготовить данные, на которых она будет учиться.

Так как в проекте будет использоваться сиамская НС для нее нужно создать отдельные папки для хранения данных.

Свертка -математическая операция, обрабатывающая изображения и другие типы данных. Она позволяет извлекать признаки и уменьшать размер изображения.

Принцип работы алгоритма свертки:

1.Выбор ядра свертки. Оно представляет собой матрицу 3×3 или 5×5 , которая будет содержать в себе весовые коэффициенты.

2.Применение ядра. Фильтр(ядро) перемещается по изображению с определенным шагом. С каждым шагом ядро поэлементно перемножается на определенную область изображения и суммируется. Результат будет представлять собой новую матрицу, которая называется выходной.

3.Применение функции активации к выходной матрице. Функции активации, применённые к выходной матрице, позволяют нейронам смоделировать нелинейные зависимости между входными и выходными данными.

В моей нейронной сети будет использован слой **Flatten**, вытягивающий матрицу в одномерный вектор, который будет подаваться на вход полносвязному слою.

Функции активации-математические функции, которые применяются для преобразования входного сигнала нейрона в выходной, также они применяются к сумме взвешенных входных сигналов и порогового значения нейрона для определения будет ли нейрон активирован.

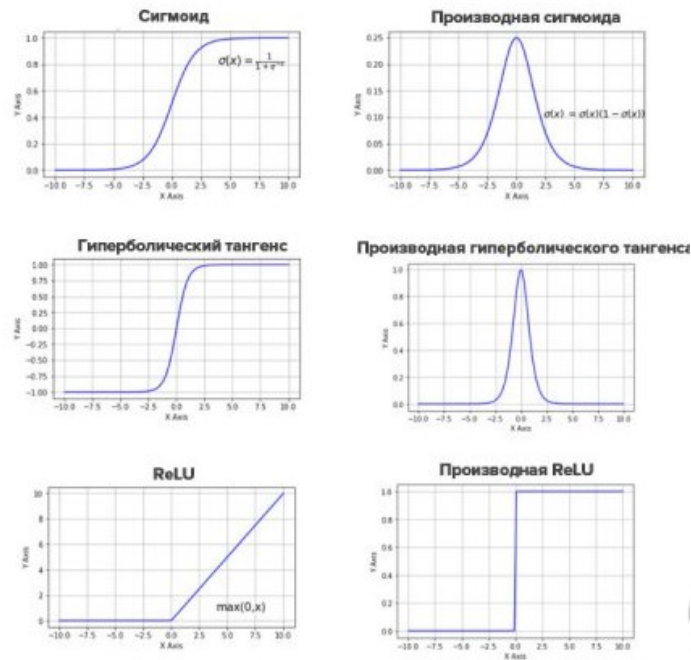


Рис. 2: Примеры функций активации

3 Практическая часть

Формулировка задачи: написать нейросеть, способную верифицировать человека по изображению.

Суть задачи заключается в том, чтоб выяснить являются ли два изображения изображениями одного и того же человека.

Сбор данных

Сбор и подготовка данных самая важная часть в создании НС. От этого будет зависеть качество ее работы. Так как для решения задачи мы будем использовать си-

амскую нейронную сеть, то для нее мы будем использовать несколько потоков данных. Сначала создадим папку 'Data', а в ней еще 3 папки 'positive', 'negative', 'anchor'.

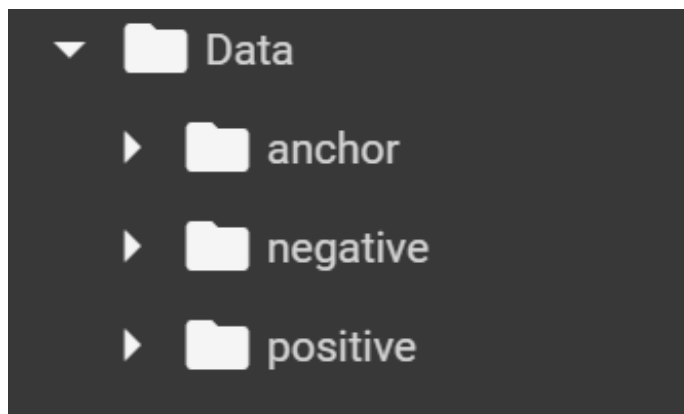


Рис. 3: Папки для хранения данных для НС

В папку 'negative' мы будем складывать фотографии других людей из датасета, скаченного с сайта [kaggle.com](https://www.kaggle.com). В коде НС реализован цикл, который перемещает все фото из нашего dataset'a в папку 'negative', главное указать путь к dataset'у. Данный датасет содержит в себе фотографии других людей в формате 250*250 пикселей. В папку 'positive' мы будем складывать изображения с веб-камеры нашего ПК. Я написал код, который активирует веб-камеру ноутбука и с помощью кнопок 'р' и 'а' будет сохранять изображения в нужном мне формате в папки 'positive' и 'anchor' соответственно. С помощи кнопки 'q' цикл прекращается, и веб-камера завершает свою работу. Фотографии, собранные с камеры также будет преобразованы в формат 250*250 пикселей.

Код по активации камеры и сбору данных в папки "positive" и "anchor". Для коллекционирования фото в 'positive' нажмите на 'p'. Для коллекционирования фото в 'anchor' нажмите на 'a'. Для завершения цикла и прекращения работы камеры нажмите на 'q'.

```

cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    frame = frame[120:120+250, 200:200+250, :]

    if cv2.waitKey(1) & 0xFF == ord('a'):
        img_name = os.path.join(ANC_PATH, '{}.jpg'.format(uuid.uuid1()))
        cv2.imwrite(img_name, frame)

    if cv2.waitKey(1) & 0xFF == ord('p'):
        img_name = os.path.join(POS_PATH, '{}.jpg'.format(uuid.uuid1()))
        cv2.imwrite(img_name, frame)

    cv2.imshow('Image Collection', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()

```

Рис. 4: Код реализации работы камеры.

Для написания НС был выбран язык программирования Python а также библиотека TensorFlow. Это один из самых популярных языков программирования, который широко используется в Machine Learning. TensorFlow- популярная и относительно простая библиотека, которая имеет в себе большое кол-во встроенных функций, позволяющая создать нейросеть любой сложности.

Импортируем все нужные нам библиотеки:

```

[ ] !pip install tensorflow==2.16.1
    !pip install keras==3.1.1

[ ] import cv2
    import shutil
    import os
    import random
    import numpy as np
    import uuid
    from matplotlib import pyplot as plt

[ ] import tensorflow as tf
    from tensorflow.keras.models import Model
    from tensorflow.keras.layers import Layer, Conv2D, Dense, MaxPooling2D, Input, Flatten

[ ] from tensorflow.keras.metrics import Precision, Recall

```

Рис. 5: Импорт библиотек.

Прописываем пути для данных, которые потом будут разделяться на тестовую и обучающую выборку.

```
POS_PATH = os.path.join('/content/drive/MyDrive/Проект HC/Data', 'positive')
NEG_PATH = os.path.join('/content/drive/MyDrive/Проект HC/Data', 'negative')
ANC_PATH = os.path.join('/content/drive/MyDrive/Проект HC/Data', 'anchor')
```

Рис. 6: Пути к папкам

После сбора данных, они нормализуются с помощью функции 'preprocess' и собираются в два набора 'positive' и 'negative'.

```
Функция обработки и нормализации фотографий.

[ ] def preprocess(file_path):
    byte_img = tf.io.read_file(file_path)
    img = tf.io.decode_jpeg(byte_img)
    img = tf.image.resize(img, (100,100))
    img = tf.math.divide(img, 255.0)
    return img

[ ] dir_path = dir_test.next()
img = preprocess(dir_path)

Объединение наборов данных в один набор.

[ ] positives = tf.data.Dataset.zip((anchor, positive, tf.data.Dataset.from_tensor_slices(tf.ones(len(anchor)))))
negatives = tf.data.Dataset.zip((anchor, negative, tf.data.Dataset.from_tensor_slices(tf.zeros(len(anchor)))))
data = positives.concatenate(negatives)
```

Рис. 7: Нормализация данных

Дальше они объединяются в одну выборку, которая кэшируется, перемешивается и разбивается на тестовую и обучающую выборки.

```
[ ] data = data.map(preprocess_twain)
    data = data.cache()
    data = data.shuffle(buffer_size=10000)

[ ] train_data = data.take(round(len(data)*.7))
    train_data = train_data.batch(16)
    train_data = train_data.prefetch(8)

[ ] test_data = data.skip(round(len(data)*.7))
    test_data = test_data.take(round(len(data)*.3))
    test_data = test_data.batch(16)
    test_data = test_data.prefetch(8)
```

Рис. 8: Создание тестовой и обучающей выборки

Разработка модели

Для нашей задачи была выбрана сиамская сверточная нейронная сеть, которая состоит из двух идентичных подсетей, поэтому стоит описать строение только одной из них.

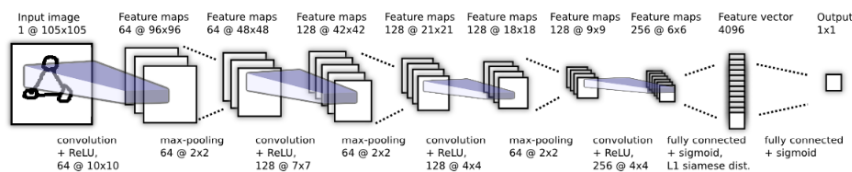


Рис. 9: Примерная структура НС

Одна из подсетей будет содержать в себе 4 сверточных слоя **Conv2D**, 3 максимизационных слоя **MaxPolling2D**. Функция активации используется **ReLU**. Размерность ядер свертки на каждом сверточном слое будет разная. На первом слое используется размерность

10*10, на втором 7*7, а на третьем и четвертом слое 4*4.

```
[ ] inp = Input(shape=(100,100,3), name='input_image')

[ ] c1 = Conv2D(64, (10,10), activation='relu')(inp)

[ ] m1 = MaxPooling2D(64, (2,2), padding='same')(c1)

[ ] c2 = Conv2D(128, (7,7), activation='relu')(m1)
    m2 = MaxPooling2D(64, (2,2), padding='same')(c2)

[ ] c3 = Conv2D(128, (4,4), activation='relu')(m2)
    m3 = MaxPooling2D(64, (2,2), padding='same')(c3)

[ ] c4 = Conv2D(256, (4,4), activation='relu')(m3)
    f1 = Flatten()(c4)
    d1 = Dense(4096, activation='sigmoid')(f1)

[ ] model = Model(inputs=[inp], outputs=[d1], name='embedding')
```

Рис. 10: Реализация НС на Python

Dense-слой нейронной сети, состоящий из одного нейрона и функции активации **sigmoid**. Он используется для вычисления вложений для каждого из входных изображений

sigmoid-математическая функция, выдающая вероятность того, что два изображения принадлежат одному и тому же классу.

Затем слой **L1Dist** вычисляет расстояние между двумя вложениями. И передает все на выходной слой, который с помощью функции активации **sigmoid** выдает вероятность того принадлежат ли два изображения одному классу.

Функция ошибки (binary cross-entropy loss), данная функция очень эффективна в задачах бинарной классификации.

Оптимизатором для НС был выбран Adam, с скоростью обучения = 0.0001, что довольно мало, но позволит добиться большой точности.

Обучение

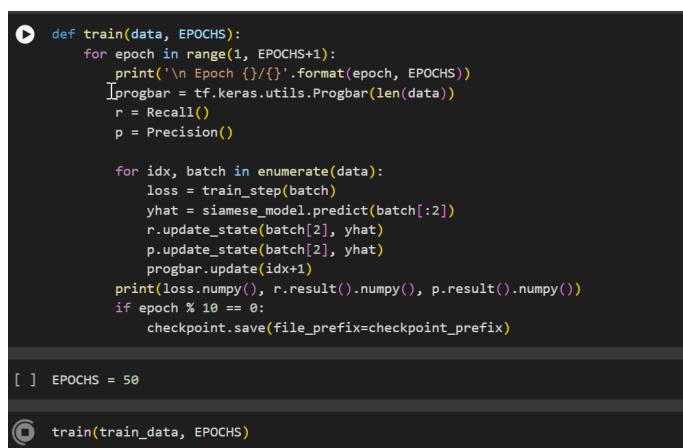
A screenshot of a code editor showing a Python function named 'train'. The function takes 'data' and 'EPOCHS' as arguments. It loops through epochs from 1 to EPOCHS+1. Inside the loop, it prints the current epoch, creates a progress bar, and initializes Recall and Precision metrics. It then iterates over the data batches. For each batch, it calls 'train_step', predicts with 'siamese_model', and updates Recall and Precision. It prints the loss, recall, and precision at the end of each epoch. If the epoch is a multiple of 10, it saves a checkpoint. At the bottom, there is a variable 'EPOCHS = 50' and a call to 'train(train_data, EPOCHS)'.

Рис. 11: Реализация обучения на Python

Обучение НС реализовано с помощью функции 'train step' которая принимает на вход batch данных. Внутри этой функции происходит обработка данных с помощью функции 'preprocess'. Затем вычисляются расстояния между вложениями. После этого вычисляются выходные значения.

Далее, вычисляется значение функции потерь с помощью функции ошибки. Затем, вычисляются градиенты функции потерь по отношению к всем обучаемым параметрам нейронной сети.

```

[49] @tf.function
def train_step(batch):
    with tf.GradientTape() as tape:
        X = batch[:2]
        y = batch[2]
        yhat = siamese_model(X, training=True)
        loss = binary_cross_loss(y, yhat)
    print(loss)
    grad = tape.gradient(loss, siamese_model.trainable_variables)
    optimizer.apply_gradients(zip(grad, siamese_model.trainable_variables))
    return loss

[50] def train(data, EPOCHS):
    for epoch in range(1, EPOCHS+1):
        print('\n Epoch {}/{}'.format(epoch, EPOCHS))
        progbar = tf.keras.utils.Progbar(len(data))
        r = Recall()
        p = Precision()

        for idx, batch in enumerate(data):
            loss = train_step(batch)
            yhat = siamese_model.predict(batch[:2])
            r.update_state(batch[2], yhat)
            p.update_state(batch[2], yhat)
            progbar.update(idx+1)
        print(loss.numpy(), r.result().numpy(), p.result().numpy())
        if epoch % 10 == 0:
            checkpoint.save(file_prefix=checkpoint_prefix)

```

Рис. 12: Реализация функций обучения на Python с использованием декоратора

```

[ ] train(train_data, EPOCHS)

10/10 ————— 436s 43s/step
0.00018529088 1.0 1.0

Epoch 47/50
1/1 ————— 4s 4s/step
1/1 ————— 4s 4s/step
1/1 ————— 4s 4s/step
1/1 ————— 4s 4s/step
1/1 ————— 4s 4s/step
1/1 ————— 4s 4s/step
1/1 ————— 6s 6s/step
1/1 ————— 4s 4s/step
1/1 ————— 4s 4s/step
1/1 ————— 4s 4s/step
10/10 ————— 430s 43s/step
4.5784923e-06 1.0 1.0

Epoch 48/50

```

Рис. 13: Процесс обучения

Проверка работаспособности НС

Давайте проверим работу НС на своих данных. С помощью схожего цикла, который мы использовали для сбора данных с веб-камеры нашего ПК. Загрузим изображение, которое будет проходить верификацию.

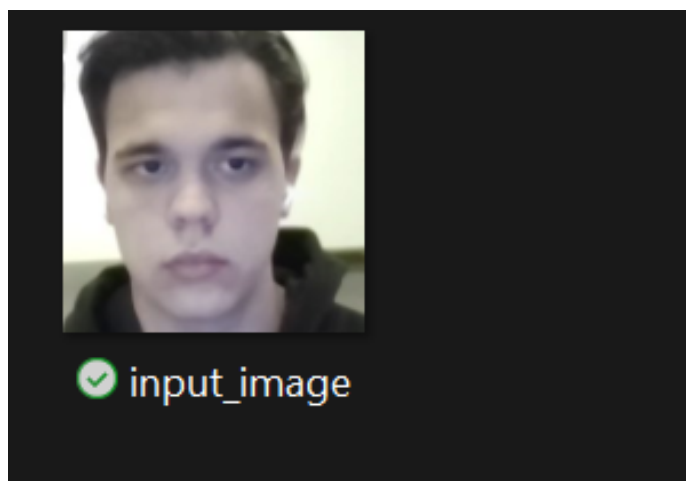


Рис. 14: Тестовое изображение

Наша НС выдаст:



Рис. 15: Результат на своем фото

Мы видим, что НС правильно распознала меня и я прошел верификацию.

Прделаем тоже самое,но для чужого человека.

Результат верификации другого человека:

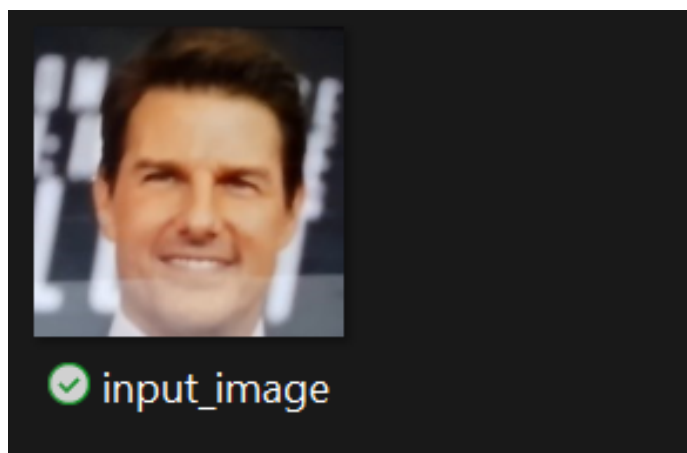


Рис. 16: Какой-то другой парень

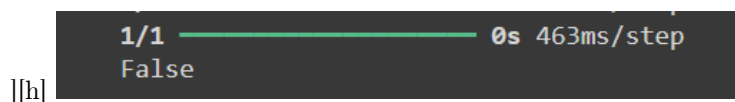


Рис. 17: Результат проверки другого парня

Как мы видим он проверку не прошел и система его дальше не пропустит.

Возможное использование и взгляд в будущее

Данная нейросеть представляет простую реализацию технологии FaceID, которая используется во многих гаджетах на сегодняшний день. В дальнейшем она может быть улучшена более сложными методами и технологиями. Также хочу отметить, что это всего лишь одна из возможностей применения сямской сверточной НС. Если взять ее за идею, но изменить немного ее суть, то она может быть применена в различных других областях. Например, определение переломов конечностей людей и животных по рентген снимкам или проверки на подлинность вещей, выставляемых на продажу.

4 Вывод

В результате работы над проектом я создал НС, которая будет верифицировать человека по фото. Нейросеть прошла тесты и показала свою работоспособность.

Это означает, что сиамские сверточные нейронные сети могут решать задачу бинарной классификации.

Данный проект в будущем может быть доработан с помощью ResNet и Inception и использоваться для проверки на подлинность, поиска дубликатов и многого другого.

5 Приложение

Таблица 1: Средние размер и количество кластеров для сети Эрдеша-Реньи с $\langle k \rangle = 8$ при различной начальной плотности σ_- (ф) и убеждаемостью γ при начале с одного кластера. При моделировании использовались 2000 различных сетей с 1000 узлов.

f	γ	Средний размер	Среднее число кластеров
0.00	5.33	1.13	0.03
	8.22	1.34	0.26
	11.11	1.55	0.70
	14.00	1.79	0.95
0.17	5.33	1.26	1.99
	8.22	13.72	5.79
	11.11	79.53	1.90
	14.00	142.30	1.16
0.33	5.33	1.48	5.26
	8.22	237.84	1.29
	11.11	319.75	1.03
	14.00	332.19	1.00
0.50	5.33	498.61	1.01
	8.22	497.94	1.01
	11.11	500.06	1.00
	14.00	500.00	1.00

$$\iiint_{-\infty}^{+\infty} \frac{1}{Z} \exp \left[-\frac{(p_x^2 + p_y^2 + p_z^2)}{2mkT} \right] dp_x dp_y dp_z = \frac{1}{Z} (2\pi mkT)^{3/2} \quad (1)$$

Формулы и таблица была выдана семинаристом Антоновым А.Д. профсеминара для полного соблюдения правил отчета

6 Список источников и мои материалы

[Книга по сиамским НС](#)

[Документация Tensorflow](#)

[Датасет LFW](#)

[Диск с материалами проекта](#)