

**ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ  
ДАННЫХ И ПОСТРОЕНИЕ  
ПРИЗНАКОВ В ЗАДАЧАХ  
ВЫЯВЛЕНИЯ НАЛИЧИЯ СОЛАНИНА  
В КЛУБНЯХ КАРТОФЕЛЯ**

**Выполнил:**

Никитин А.Д.

группа ПМ21-2

**Научный руководитель:**

к.т.н., доцент

Моисеев Г. В.



# Постановка задачи

- 1) Предварительный сбор данных (составление датасета) из изображений, разделение их по видам.
- 2) Создание набора данных для тестирования.
- 3) Создание нейронной сети.
- 4) Компиляция модели и ее обучение.
- 5) Оценка качества обучения и проверка.



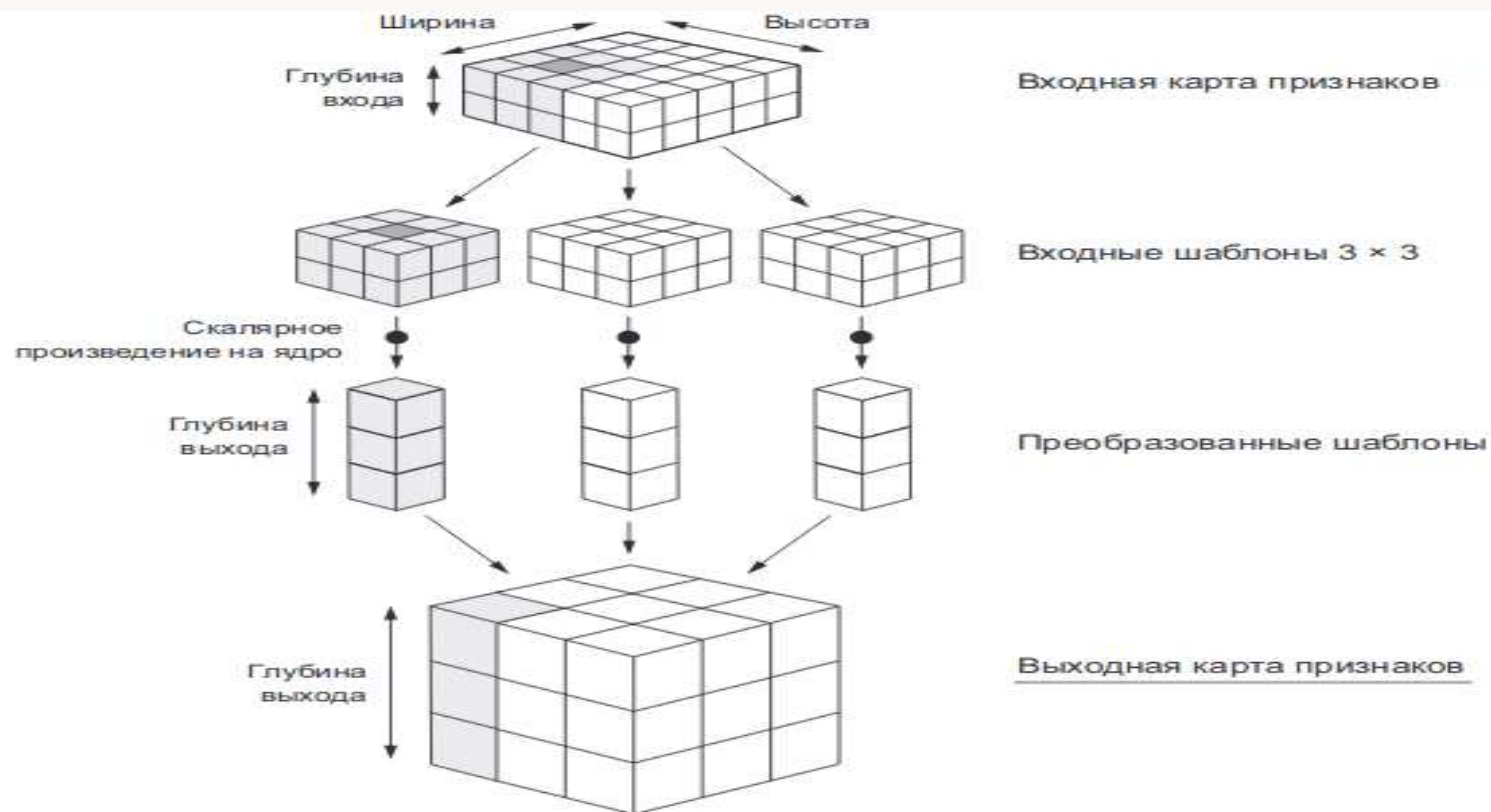
Библиотеки для  
компьютерного зрения



TensorFlow

# Метод распознавания изображений

Convolutional neural network (CNN, ConvNet), или Сверточная нейронная сеть — класс глубоких нейронных сетей, часто применяемый в анализе визуальных образов.



Картинка разбивается на маленькие участки, вплоть до нескольких пикселей, каждый из которых будет входным нейроном. С помощью синапсов сигналы передаются от одного слоя к другому. Во время этого процесса сотни тысяч нейронов с миллионами параметров сравнивают полученные сигналы с уже обработанными данными.

Проще говоря, если мы просим машину распознать фотографию картошки, мы разобьем фото на маленькие кусочки и будем сравнивать эти слои с миллионами уже имеющихся изображений картошки, значения признаков которых сеть выучила.



# Описание датасета

Всего: 216 изображений. Из них:



По 108 изображений картофеля с солянином и без



# ПРЕДОБРАБОТКА ДАННЫХ

Файлы

↑

↶

↷

🗑

{x}

..

▶ data

▶ sample\_data

▼ test

▶ Healthy\_potato

▶ Potato\_with\_solanine

▼ train

▶ Healthy\_potato

▶ Potato\_with\_solanine

Набор данных для обучения

```
[10] train_dataset = image_dataset_from_directory('train',
                                                subset='training',
                                                seed=42,
                                                validation_split=0.1,
                                                batch_size=batch_size,
                                                image_size=image_size)

Found 180 files belonging to 2 classes.
Using 162 files for training.
```

Проверочный набор данных

```
[11] validation_dataset = image_dataset_from_directory('train',
                                                    subset='validation',
                                                    seed=42,
                                                    validation_split=0.1,
                                                    batch_size=batch_size,
                                                    image_size=image_size)

Found 180 files belonging to 2 classes.
Using 18 files for validation.
```

Названия классов в наборах данных.


```
[12] class_names = train_dataset.class_names
     class_names

['Healthy_potato', 'Potato_with_solanine']
```


Примеры изображений

```
[13] plt.figure(figsize=(8, 8))
     for images, labels in train_dataset.take(1):
         for i in range(9):
             ax = plt.subplot(3, 3, i + 1)
             plt.imshow(images[i].numpy().astype("uint8"))
             plt.title(class_names[labels[i]])
             plt.axis("off")
```


Healthy\_potato




Potato\_with\_solanine




Potato\_with\_solanine




Potato\_with\_solanine



Potato\_with\_solanine



Healthy\_potato



# СОЗДАНИЕ НЕЙРОННОЙ СЕТИ

## ▼ Настраиваем производительность TensorFlow DataSet'ов

```
✓ [i] AUTOTUNE = tf.data.experimental.AUTOTUNE

train_dataset = train_dataset.prefetch(buffer_size=AUTOTUNE)
validation_dataset = validation_dataset.prefetch(buffer_size=AUTOTUNE)
test_dataset = test_dataset.prefetch(buffer_size=AUTOTUNE)
```

## ▼ Создаем нейронную сеть

```
▶ # Создаем последовательную модель
model = Sequential()
# Сверточный слой
model.add(Conv2D(16, (5, 5), padding='same',
                 input_shape=(100, 100, 3), activation='relu'))
# Слой подвыборки
model.add(MaxPooling2D(pool_size=(2, 2)))
# Сверточный слой
model.add(Conv2D(32, (5, 5), activation='relu', padding='same'))
# Слой подвыборки
model.add(MaxPooling2D(pool_size=(2, 2)))
# Сверточный слой
model.add(Conv2D(64, (5, 5), activation='relu', padding='same'))
# Слой подвыборки
model.add(MaxPooling2D(pool_size=(2, 2)))
# Сверточный слой
model.add(Conv2D(128, (5, 5), activation='relu', padding='same'))
# Слой подвыборки
model.add(MaxPooling2D(pool_size=(2, 2)))
# Полносвязная часть нейронной сети для классификации
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
# Выходной слой, 2 нейрона по количеству классов
model.add(Dense(2, activation='softmax'))
```



# ОБУЧЕНИЕ И ОЦЕНКА КАЧЕСТВА ОБУЧЕНИЯ НЕЙРОННОЙ СЕТИ

```
[19] history = model.fit(train_dataset,
                        validation_data=validation_dataset,
                        epochs=15,
                        verbose=2)

Epoch 1/15
WARNING:tensorflow: out of the last 5 calls to function BaseOptimizer.update_step via at 0x7f6f182df25a triggered tf.function retracing. Tracing is expensive and the excessive num
WARNING:tensorflow: out of the last 5 calls to function BaseOptimizer.update_step via at 0x7f6f182df25a triggered tf.function retracing. Tracing is expensive and the excessive num
11/11 - 15s - loss: 0.71207 - accuracy: 0.4015 - val_loss: 0.6013 - val_accuracy: 0.7222 - 15s/epoch - 1s/step
Epoch 2/15
11/11 - 1s - loss: 0.7057 - accuracy: 0.5617 - val_loss: 0.8441 - val_accuracy: 0.3809 - 1s/epoch - 97ms/step
Epoch 3/15
11/11 - 2s - loss: 0.6707 - accuracy: 0.5494 - val_loss: 0.6572 - val_accuracy: 0.6667 - 2s/epoch - 141ms/step
Epoch 4/15
11/11 - 1s - loss: 0.6794 - accuracy: 0.5242 - val_loss: 0.6791 - val_accuracy: 0.3809 - 1s/epoch - 95ms/step
Epoch 5/15
11/11 - 1s - loss: 0.6581 - accuracy: 0.6235 - val_loss: 0.6197 - val_accuracy: 0.7222 - 1s/epoch - 100ms/step
Epoch 6/15
11/11 - 1s - loss: 0.6199 - accuracy: 0.6790 - val_loss: 0.6079 - val_accuracy: 0.5000 - 1s/epoch - 99ms/step
Epoch 7/15
11/11 - 1s - loss: 0.5755 - accuracy: 0.7054 - val_loss: 0.6291 - val_accuracy: 0.3809 - 1s/epoch - 98ms/step
Epoch 8/15
11/11 - 1s - loss: 0.5583 - accuracy: 0.6802 - val_loss: 0.5687 - val_accuracy: 0.7222 - 1s/epoch - 93ms/step
Epoch 9/15
11/11 - 1s - loss: 0.4039 - accuracy: 0.8500 - val_loss: 0.4182 - val_accuracy: 0.7778 - 1s/epoch - 92ms/step
Epoch 10/15
11/11 - 1s - loss: 0.3267 - accuracy: 0.8827 - val_loss: 0.4552 - val_accuracy: 0.8333 - 1s/epoch - 98ms/step
Epoch 11/15
11/11 - 1s - loss: 0.3073 - accuracy: 0.8951 - val_loss: 0.3794 - val_accuracy: 0.8333 - 1s/epoch - 90ms/step
Epoch 12/15
11/11 - 1s - loss: 0.3548 - accuracy: 0.8837 - val_loss: 0.2897 - val_accuracy: 0.7778 - 1s/epoch - 98ms/step
Epoch 13/15
11/11 - 2s - loss: 0.3102 - accuracy: 0.8889 - val_loss: 0.0832 - val_accuracy: 0.8889 - 2s/epoch - 159ms/step
Epoch 14/15
11/11 - 1s - loss: 0.2135 - accuracy: 0.9130 - val_loss: 0.0350 - val_accuracy: 0.6667 - 1s/epoch - 121ms/step
Epoch 15/15
11/11 - 1s - loss: 0.3005 - accuracy: 0.8642 - val_loss: 0.2508 - val_accuracy: 0.8889 - 1s/epoch - 96ms/step
```

Обучаем модель

## Компилируем модель

```
✓ [18] model.compile(loss='sparse_categorical_crossentropy',
0      optimizer="adam",
ек.    metrics=['accuracy'],
      run_eagerly=True)
```

## Оцениваем качество обучения сети

```
✓ [20] # Оцениваем качество обучения модели на тестовых данных
0      scores = model.evaluate(test_dataset, verbose=1)
ек.
```

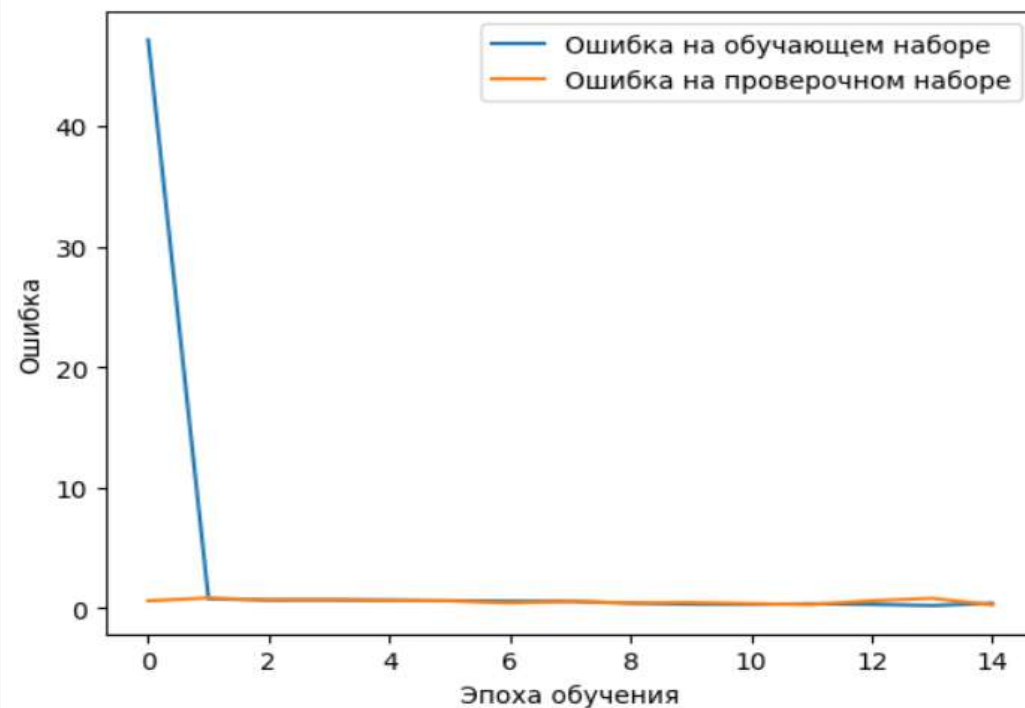
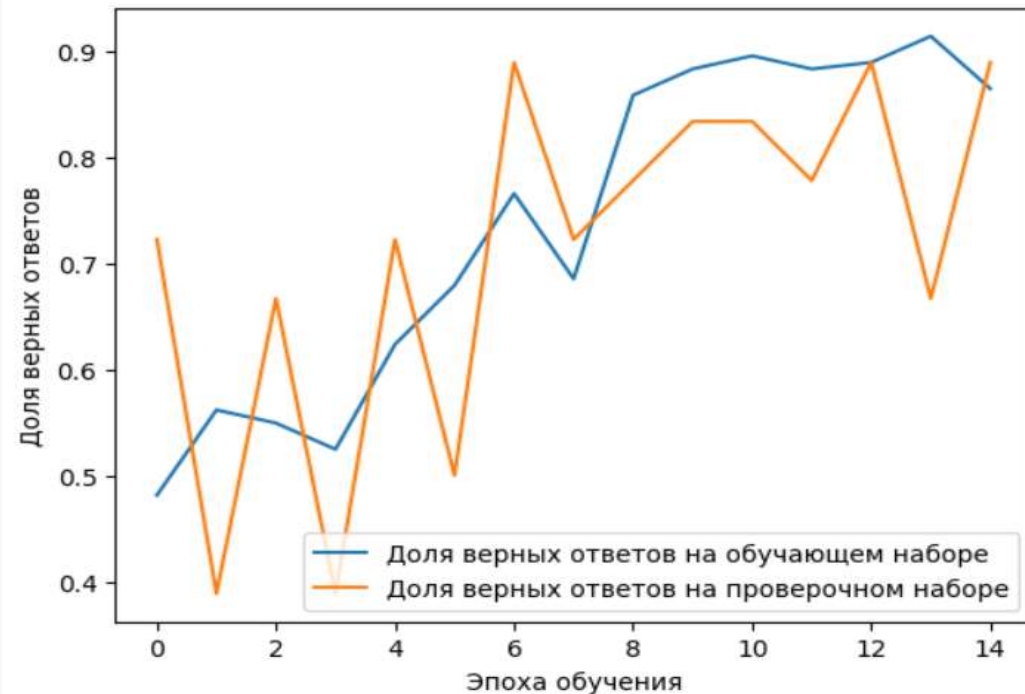
3/3 [=====] - 0s 23ms/step - loss: 0.3347 - accuracy: 0.8824

```
✓ [21] print("Доля верных ответов на тестовых данных, в процентах:", round(scores[1] * 100, 4))
0
ек.
```

Доля верных ответов на тестовых данных, в процентах: 88.2353

Итог: 88% верных ответов

# ГРАФИКИ ПРАВИЛЬНЫХ ОТВЕТОВ И ОШИБКИ НА ОБУЧАЮЩЕМ И ПРОВЕРОЧНОМ НАБОРЕ ДАННЫХ.



# Использование готовой нейронной сети для распознавания изображений.

```
[26] prediction = model.predict(test_dataset)

1/3 [-----] - 0s 13ms/step

[28] for elem in prediction:
      predictionz = np.argmax(elem)
      print("Номер класса:", predictionz)
      print("Название класса:", classes[predictionz])

Номер класса: 1
Название класса: Potato_with_solanine
Номер класса: 0
Название класса: Healthy_potato
Номер класса: 1
Название класса: Potato_with_solanine
Номер класса: 0
Название класса: Healthy_potato
Номер класса: 1
Название класса: Potato_with_solanine
Номер класса: 1
Название класса: Potato_with_solanine
Номер класса: 0
Название класса: Healthy_potato


Выберем случайное изображение и проверим

[29] data_sample = next(iter(test_dataset))
      sample_image = data_sample[0].numpy()[0]
      sample_label = classes[data_sample[1].numpy()[0]]
      prediction = np.argmax(model.predict(sample_image.reshape(-1, *sample_image.shape))[0])
      print("Номер класса:", prediction)
      print("Название класса:", classes[prediction])

1/1 [=====] - 0s 158ms/step
Номер класса: 1
Название класса: Potato_with_solanine

plt.figure(figsize=(4, 4))
plt.imshow(data_sample[0].numpy()[0].astype("uint8"))
plt.title(classes[prediction])
plt.axis("off")

(-0.5, 99.5, 99.5, -0.5)
Potato_with_solanine
```



# **Анализ результатов нейросети в различных условиях**



**Результат работы  
нейросети на 10  
изображениях в  
датасете:**

**85%**

**Верных  
ответов**



**Результат работы  
нейросети на 10  
изображениях в  
датасете:**

**75%**

**Верных  
ответов**



# Заключение

Точность нейросети:

1. На тестовых данных: 88%
2. На данных с перекрытием листвой: 85%
3. На данных с грязным картофелем: 75%



# Список используемой литературы

1. Автор: Andrey Sozykin. Канал на YouTube курс “Программирование глубоких нейронных сетей”, URL: <https://www.youtube.com/@AndreySozykin>
2. Автор: К ВВ. Как создать классификатор изображений на Python с помощью Tensorflow 2 и Keras, URL: [waksoft.susu.ru](http://waksoft.susu.ru)
3. Автор: Adrian Rosebrock, September 10, 2018. Keras Tutorial: How to get started with Keras, Deep Learning, and Python, URL: <https://www.reg.ru/blog/keras/amp/>
4. GitHub, URL: [https://github.com/sozykin/dlpython\\_course](https://github.com/sozykin/dlpython_course)
5. Автор: Evgenii Legotckoi, 13 мая 2020. Распознавание изображений на Python с помощью TensorFlow и Keras, URL: <https://evileg.com/ru/post/619/>
6. Автор: bredd\_owen, 14 фев 2017. Создаём нейронную сеть InceptionV3 для распознавания изображений URL: <https://habr.com/ru/articles/321834/>