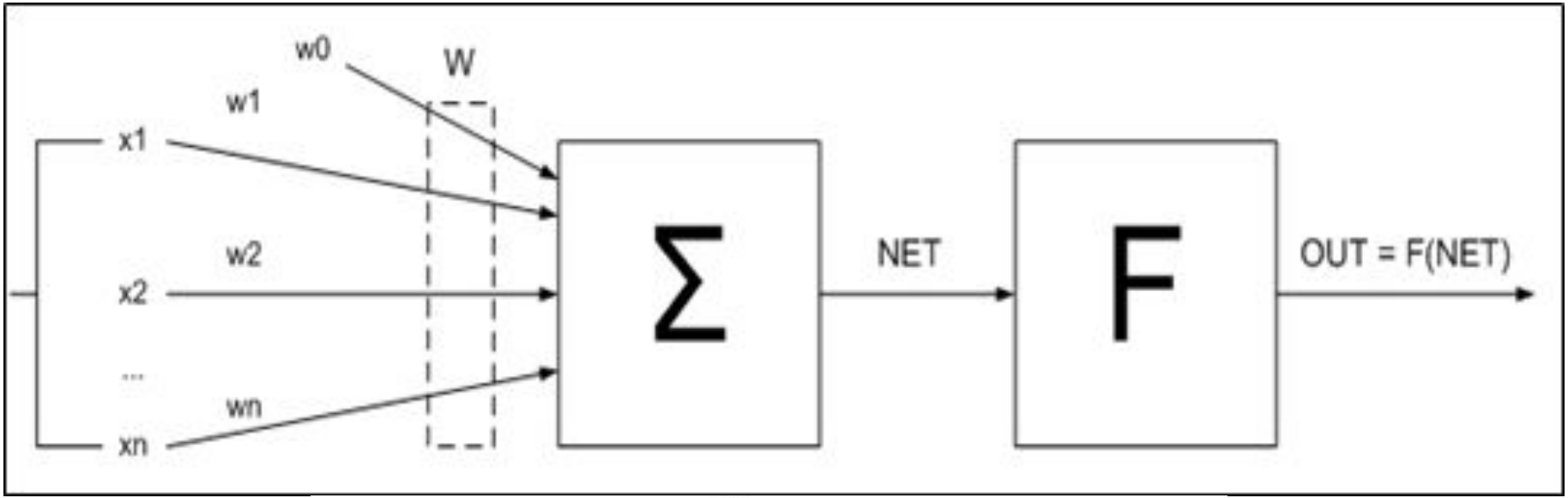
Нейронные сети глубокого обучения



$$NET = \sum_{i=1}^{n} w_i * x_i + w_0 \ ,$$

где w_i – вес i нейрона;

x_i – выход i нейрона;

w_0 – вспомогательный параметр, смещение;

n – количество синаптических связей, входящих в нейрон.

Задача

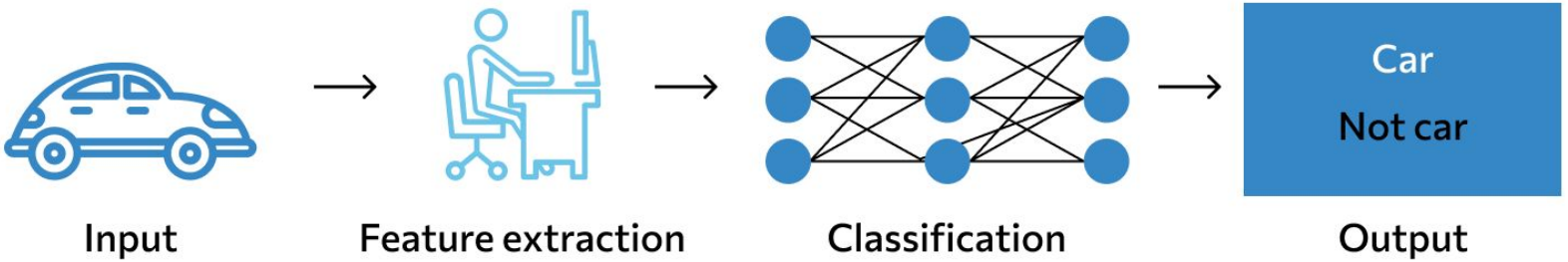
Задача классификации изображений — это приём начального изображения и вывод его класса (кошка, собака и т.д.) или группы вероятных классов, которая лучше всего характеризует изображение.

What We See

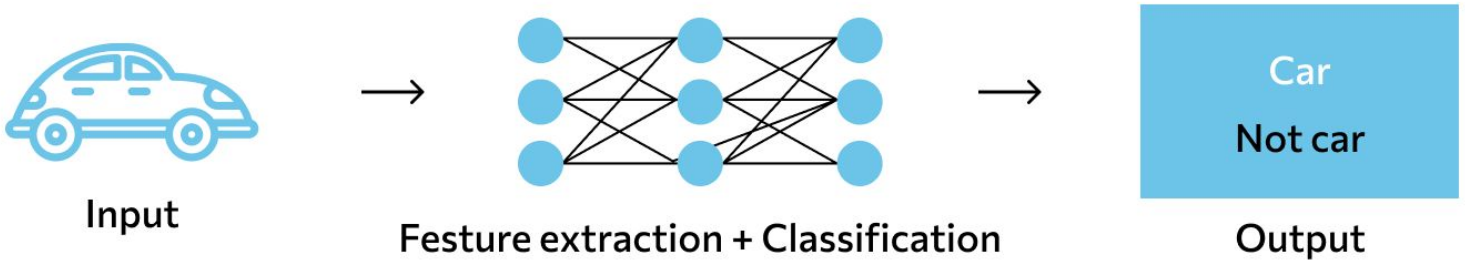
What Computers See

Идея в том, что вы даете компьютеру эту матрицу, а он выводит числа, которые описывают вероятность класса изображения (.80 для кошки, .15 для собаки, .05 для птицы и т.д.).

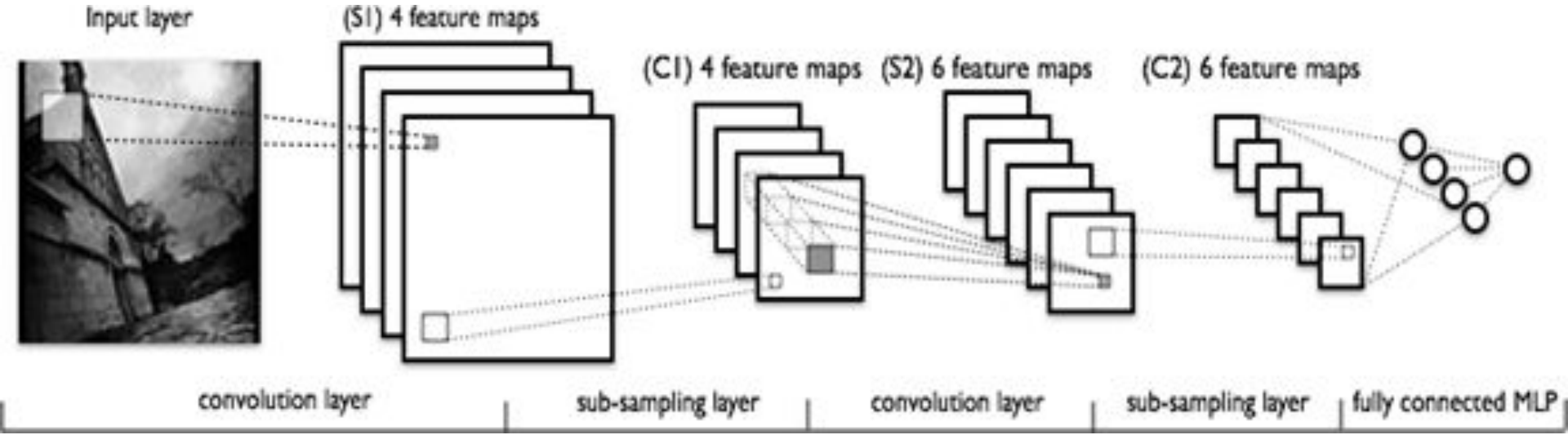
Machine Learning

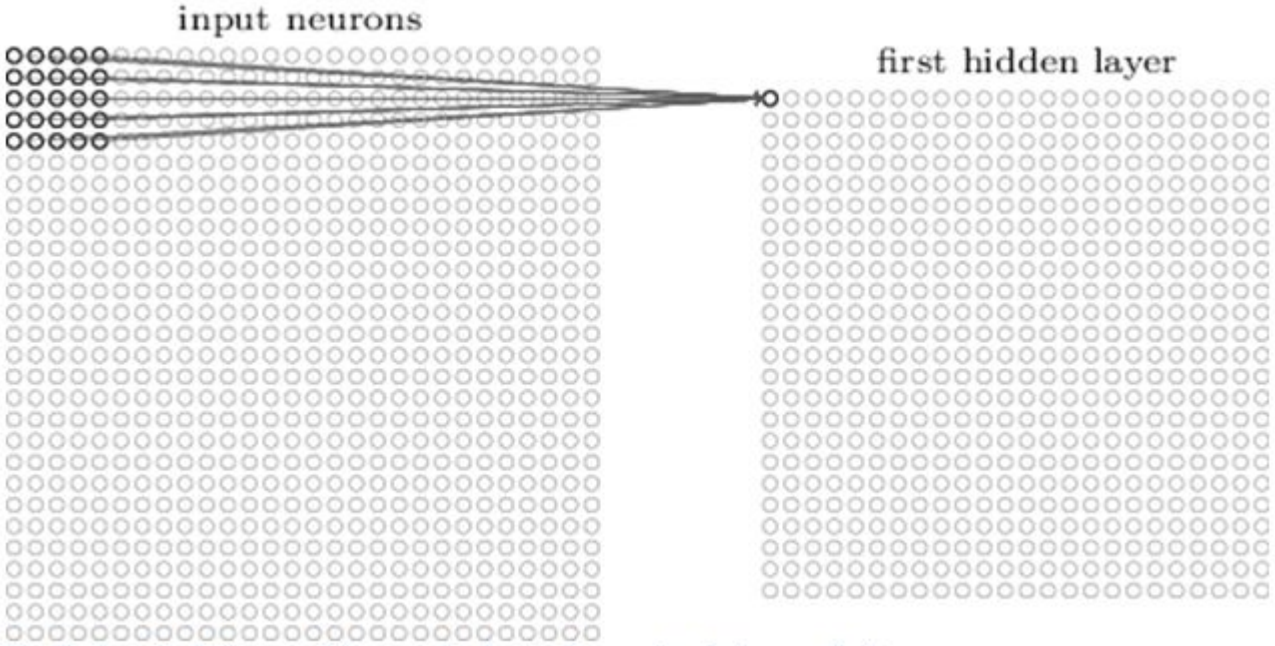


Deep Learning



СНС состоит из разных видов слоев: сверточные (convolutional) слои, субдискретизирующие (subsampling, подвыборка) слои и слои «обычной» нейронной сети – персептрона



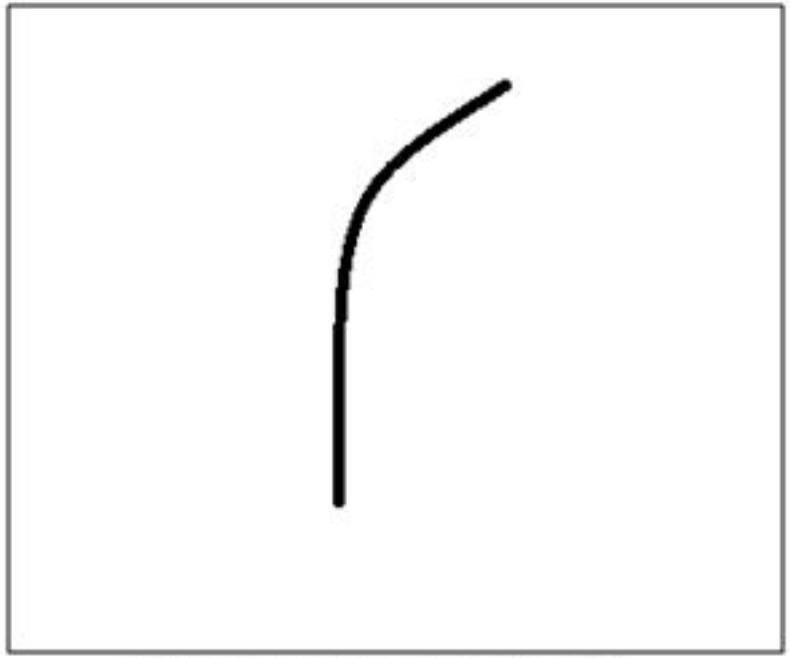


Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

Фильтр производит свёртку, то есть передвигается по вводному изображению, он умножает значения фильтра на исходные значения пикселей изображения (поэлементное умножение). Все эти умножения суммируются. И в итоге получается одно число.

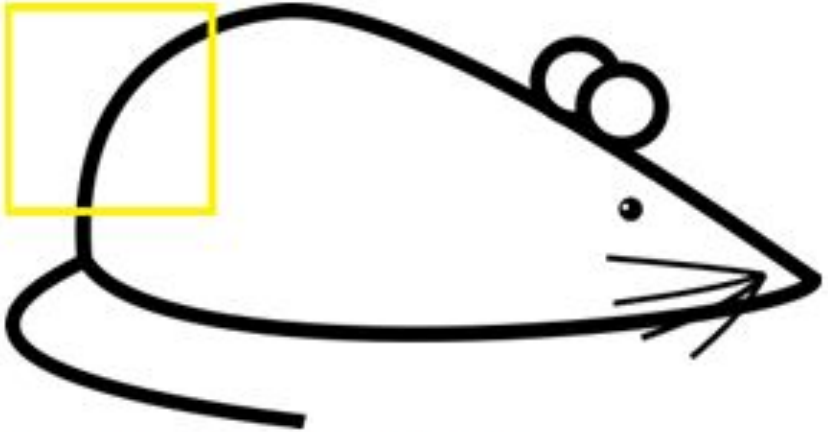
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

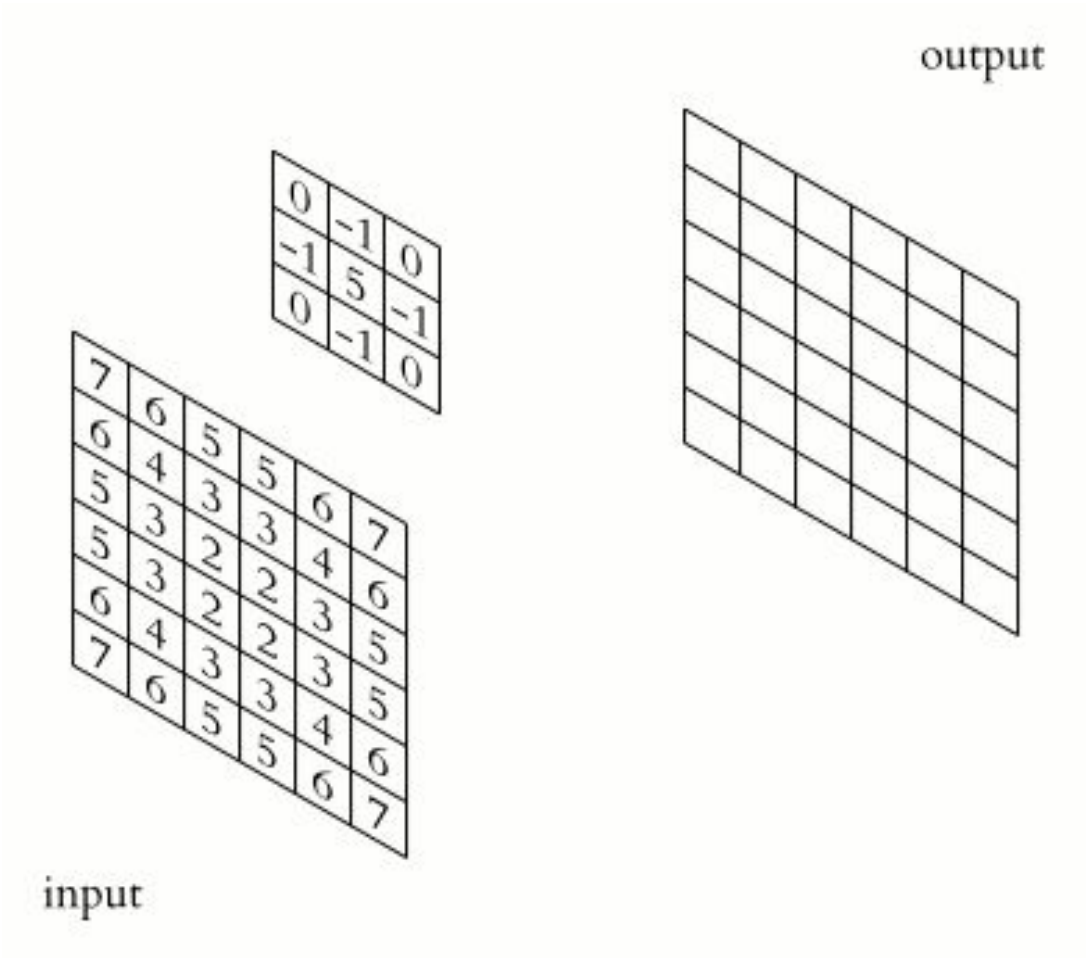
Pixel representation of receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

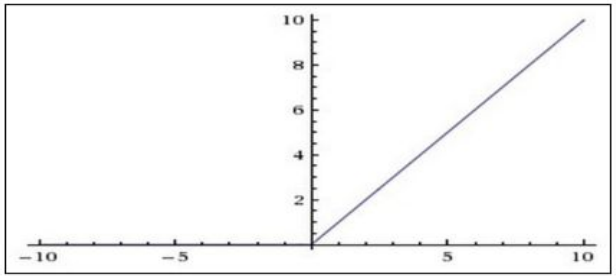
Multiplication and Summation = 0


output
0 -1 0
-1 5 -1
0 -1 0
7 6 5 5 6 7
6 4 3 3 4 6
5 3 2 2 3 5
5 3 2 2 3 5
6 4 3 3 4 6
7 6 5 5 6 7
input

Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool ->Fully Connected

Слой RELU (блок линейной ректификации) применяет поэлементную функцию активации вроде f (x) = max(0,x), устанавливая нулевой порог. Иными словами, RELU выполняет следующие действия: если x > 0, то объем остается прежним ([32×32×12]), а если x < 0, то осекаются ненужные детали в канале и путем замены на 0.

ReLU (rectified linear unit)



$$f(s) = \max(0, s)$$

$$f'(s) = \begin{cases} 1, & s > 0 \\ rand(0.01, 0.05), & s \leq 0 \end{cases}$$

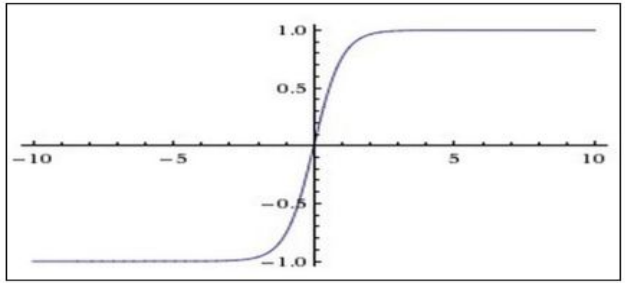
Достоинства

- лишена ресурсоемких операций
- отсекает ненужные детали
- отсутствует разрастание/затухание градиента
- быстрое обучение

Недостатки

- не всегда надежна, в процессе обучения может “умирать”
- сильно зависима от инициализации весов

Гиперболический тангенс



$$f(s) = \frac{e^{2s} - 1}{e^{2s} + 1}$$

$$f'(s) = 1 - f(s)^2$$

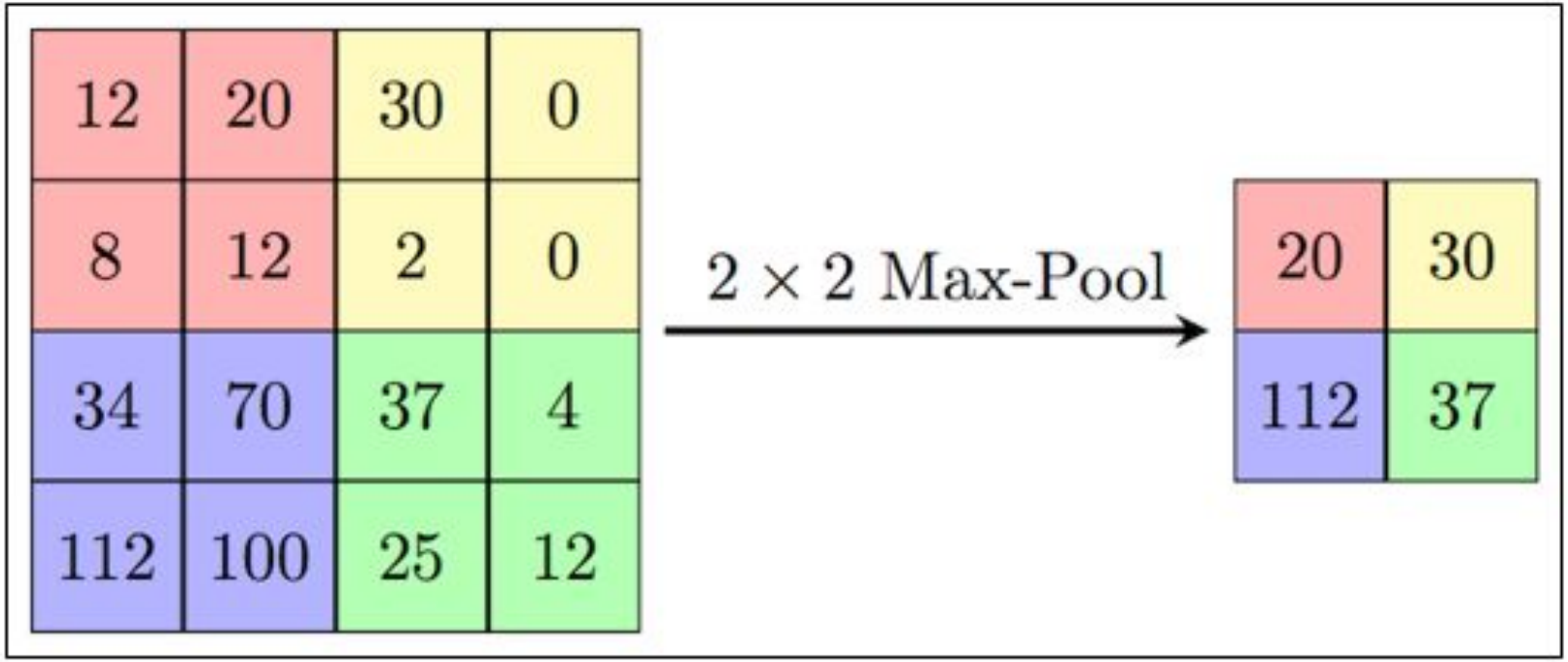
Достоинства

- простое вычисление производной через значение своей функции
- область значений от -1 до 1

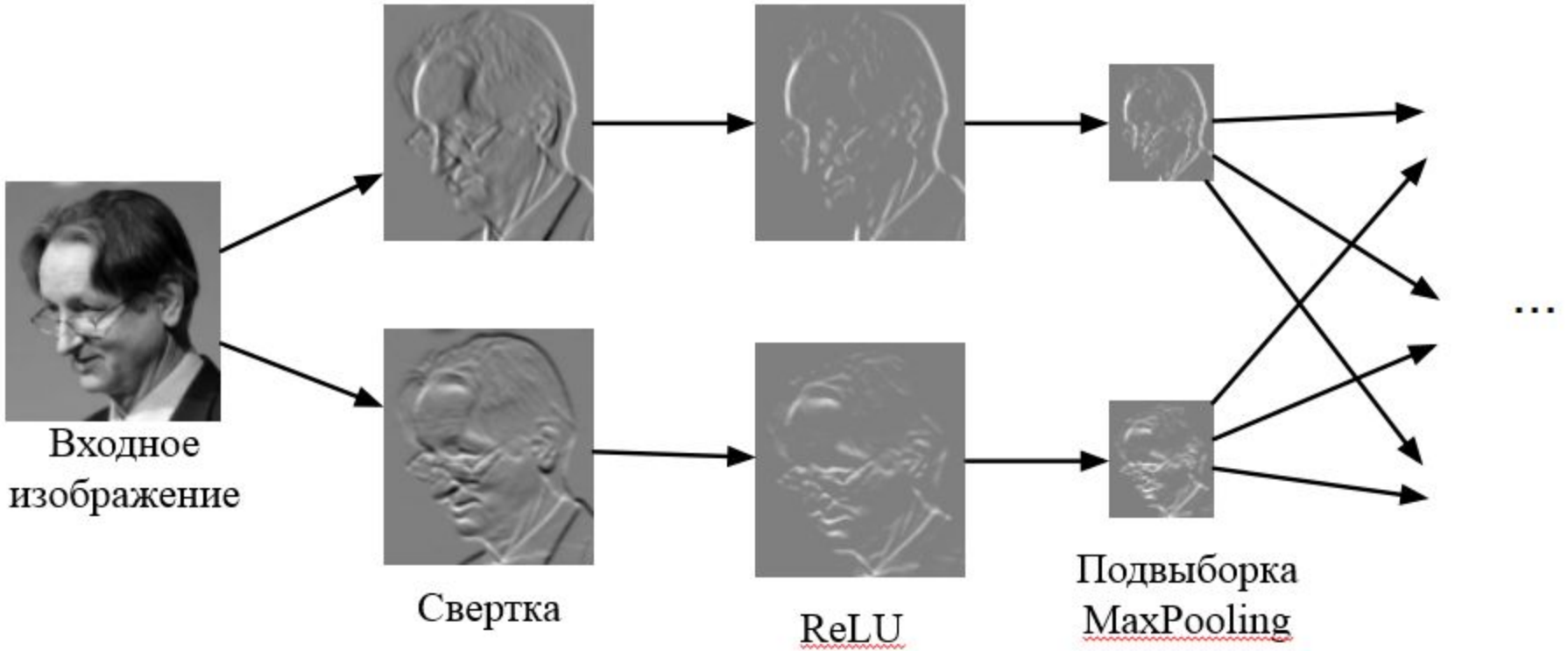
Недостатки

- затухание или увеличение градиента
- ресурсоемкая по сравнению с ReLU

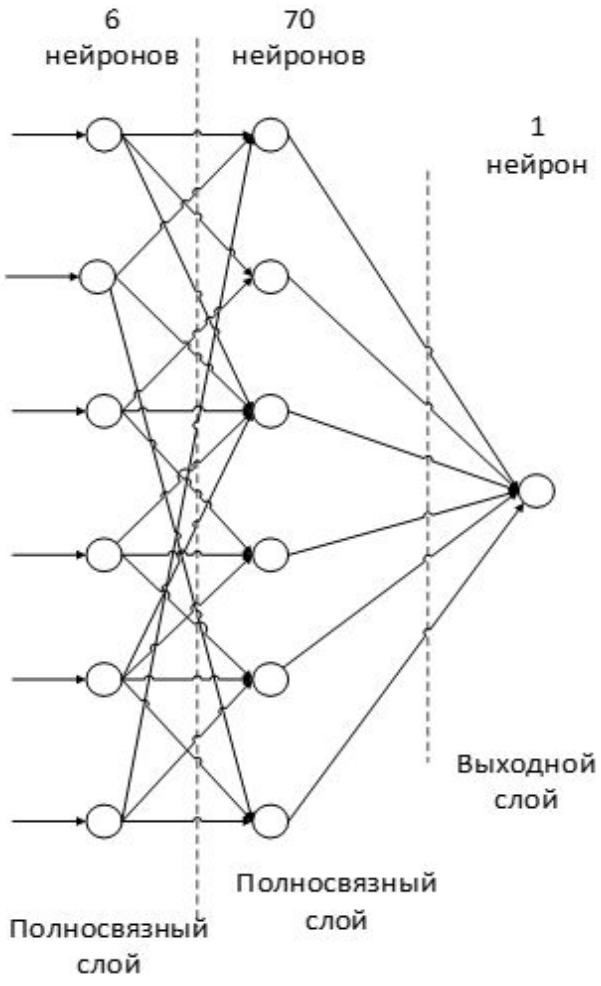
Слой POOL (слой пулинга) выполняет операцию по понижающей дискретизации пространственных размеров (ширина и высота), в результате чего объем может сократиться до [16×16×12]. То есть на этом этапе выполняется нелинейное уплотнение карты признаков. Логика работы такова: если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробной картинки.


12 20 30 0
8 12 2 0
34 70 37 4
112 100 25 12
2 × 2 Max-Pool
20 30
112 37

Операция свертки + ReLU + подвыборка



Слой FC (полносвязный слой) выводит N-мерный вектор (N — число классов) для определения нужного класса. Работа организуется путем обращения к выходу предыдущего слоя (карте признаков) и определения свойств, которые наиболее характерны для определенного класса.

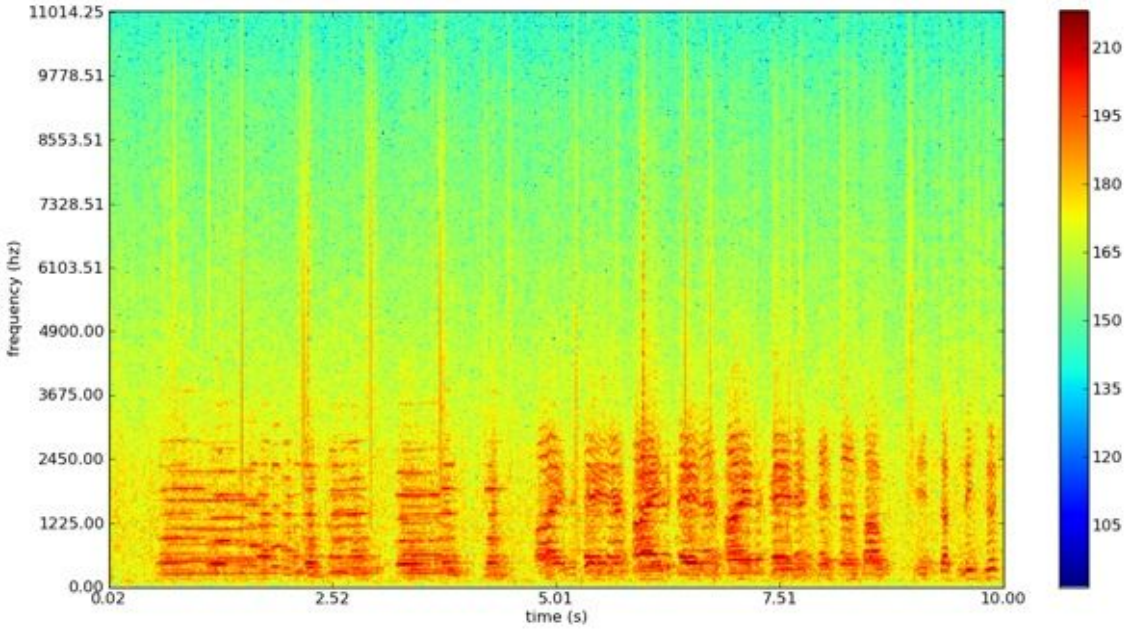

6
нейронов
70
нейронов
1
нейрон
Выходной
слой
Полносвязный
слой
Полносвязный
слой

Spoken language identification with deep convolutional networks

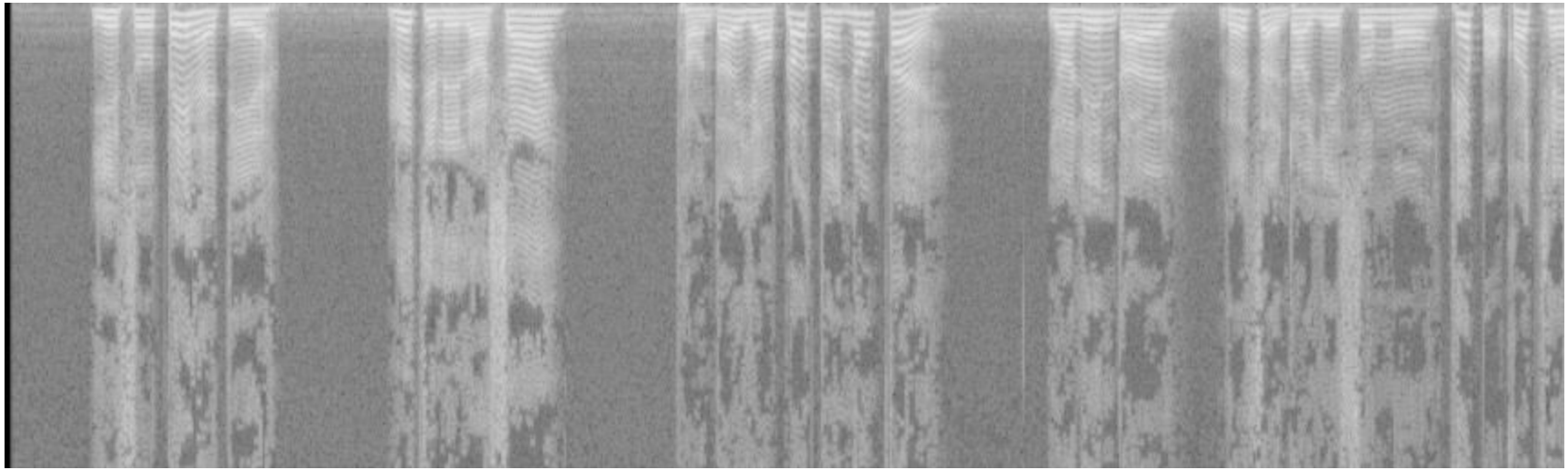
http://yerevann.github.io/2015/10/11/spoken-language-identification-with-deep-convolutional-networks/

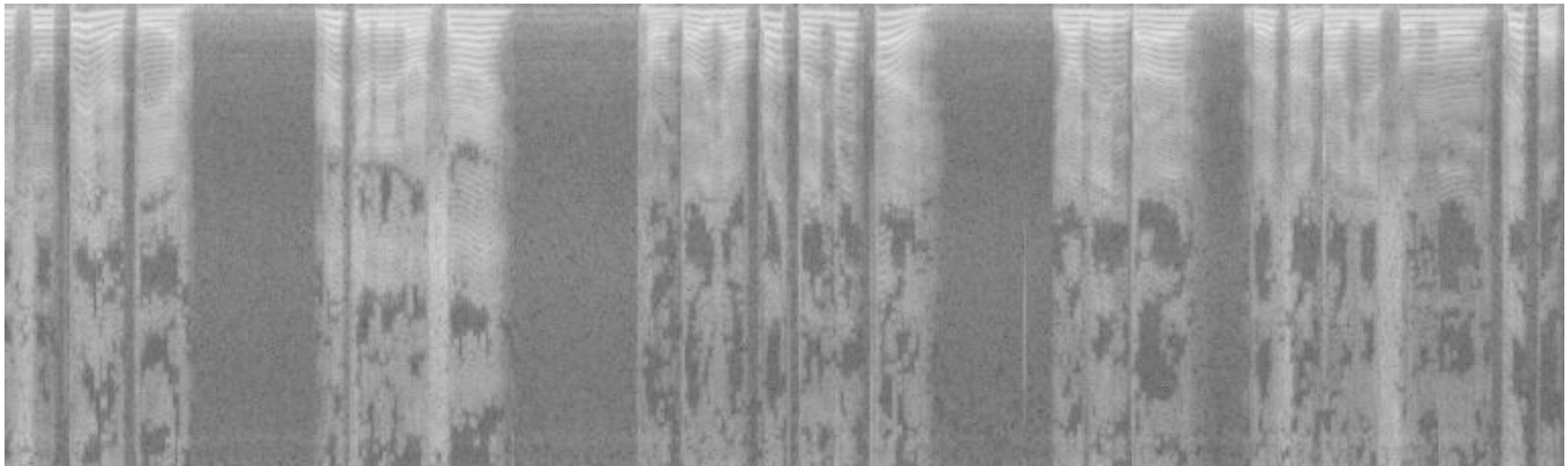
Спектрограмма из `wav` файла: http://www.frank-zalkow.de/en/code-snippets/create-audio-spectrograms-with-python.html?ckattempt=1&i=1

Рандомно разрезанные спектрограммы: https://github.com/YerevaNN/Spoken-language-identification/blob/master/augment_data.py#L77



https://github.com/YerevaNN/Spoken-language-identification



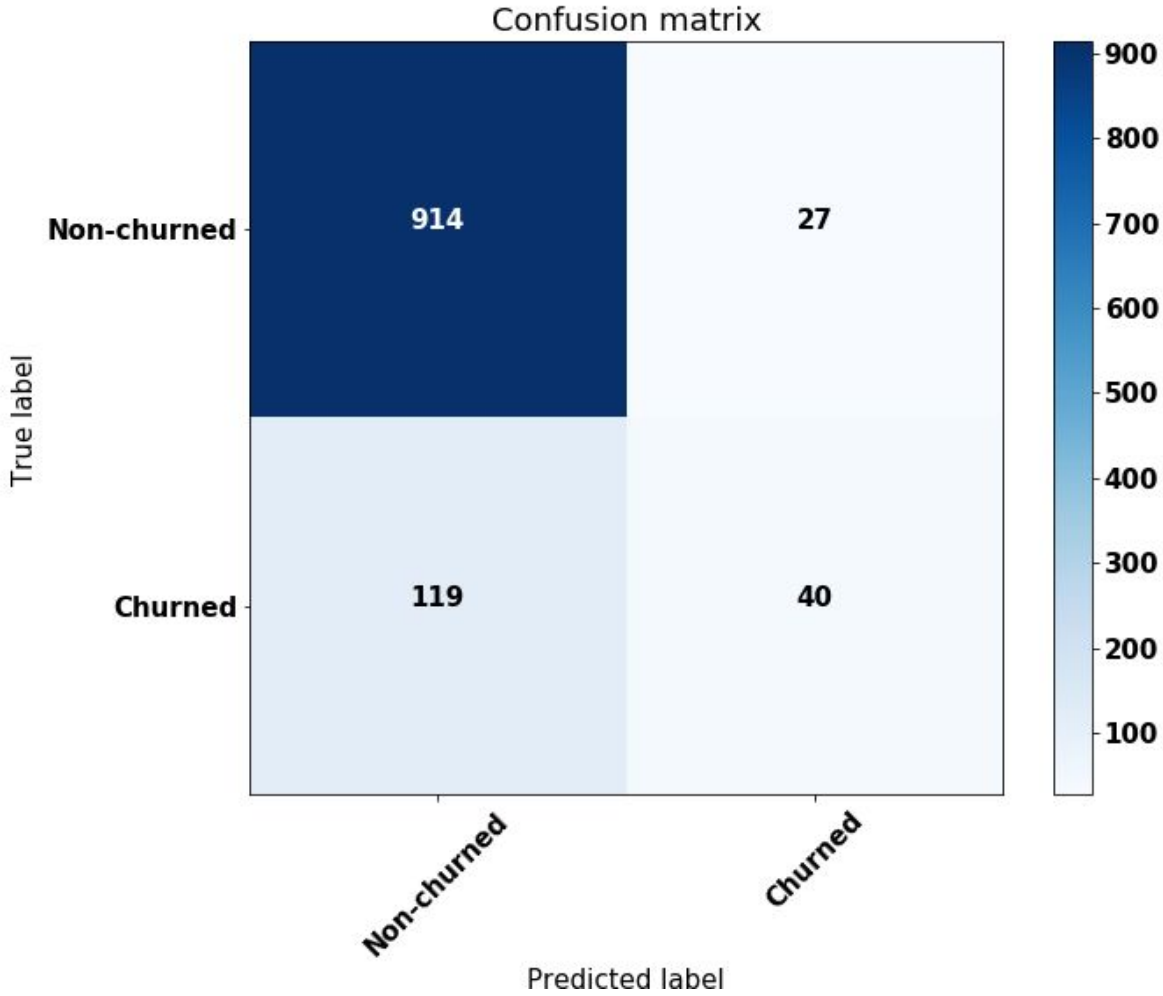


Accuracy, precision и recall

Допустим, что у нас есть два класса и алгоритм, предсказывающий принадлежность каждого объекта одному из классов, тогда матрица ошибок классификации будет выглядеть следующим образом:

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Здесь $\hat{y}$ — это ответ алгоритма на объекте, а y — истинная метка класса на этом объекте. Таким образом, ошибки классификации бывают двух видов: False Negative (FN) и False Positive (FP).


Confusion matrix
914
27
119
40
Non-churned
Churned
True label
Non-churned
Churned
Predicted label
900
800
700
600
500
400
300
200
100

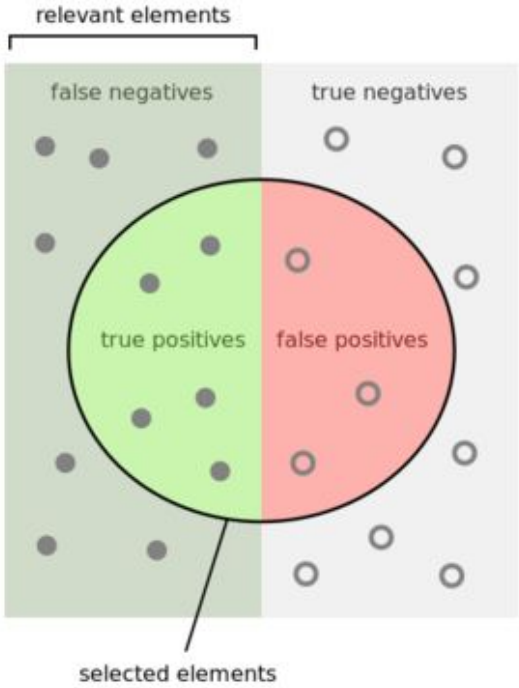
- Accuracy
  Интуитивно понятной, очевидной и почти неиспользуемой метрикой является accuracy — доля правильных ответов алгоритма:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а **recall** показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

Метрика Accuracy

- Accuracy — это показатель, который описывает общую точность предсказания модели по всем классам. Это особенно полезно, когда каждый класс одинаково важен. Он рассчитывается как отношение количества правильных прогнозов к их общему количеству.

Keras

https://keras.io/

Keras — это библиотека для языка программирования Python, которая предназначена для глубокого машинного обучения. Она позволяет быстрее создавать и настраивать модели — схемы, по которым распространяется и подсчитывается информация при обучении. Но сложных математических вычислений Keras не выполняет и используется как надстройка над другими библиотеками.

Для чего нужен Keras

- Удобное построение моделей, по которым будет проходить обучение.
- Настройка слоев в моделях — обычно подбор нужного количества слоев требуется для точности.
- Обработка ввода и вывода информации из модели.
- Преобразование входных данных, которые поступают в обучаемую модель.
- Удобный подбор датасетов для обучения.
- Визуализация модели.
- Подготовка модели к работе, определение ее функций ошибки и оптимизаторов.
- Обучение и тестирование модели.
- Сборка и первичный запуск программы машинного обучения

Особенности Keras

- Написана на чистом Python, чтобы код был понятнее и легче поддерживался.
- Работает на большинстве существующих платформ: не только в ОС Windows и Linux, но и на микрокомпьютерах, мобильных устройствах, в облаке или в браузере.
- Поддерживает работу с CPU и GPU — с обычным или графическим процессором. Последний часто используют для сложных вычислений, когда CPU не справляется.
- Поддерживает разные виды нейронных сетей: классические перцептроны, сверточные и рекуррентные сети, их комбинации.
- Совместима с Python начиная с версии 2.7 вплоть до современных.

pip install keras

1. TensorFlow
2. NumPy — это библиотека Python, которую применяют для математических вычислений: начиная с базовых функций и заканчивая линейной алгеброй. Полное название библиотеки — Numerical Python extensions, или «Числовые расширения Python».
3. SciPy — это библиотека для языка Python, основанная на расширении NumPy, но для более глубоких и сложных научных вычислений, анализа данных и построения графиков. SciPy в основном написана на Python и частично на языках C, C++ и Fortran, поэтому отличается высокой производительностью и скоростью работы.

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, BatchNormalization, Activation, Flatten, Dropout

# полносвязная сеть
model = Sequential()
model.add(Dense(8, activation='relu', input_dim=8))
model.add(Dropout(0.1))
model.add(Dense(20, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation='linear'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=30, epochs=50, verbose=1)

print(model.summary())
```

```
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.fit(X, y, epochs=50, batch_size=30)

_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100))
```

```
# полносвязная сеть
model = Sequential()
model.add(Dense(8, activation='relu', input_dim=8))
model.add(Dropout(0.1))
model.add(Dense(20, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation='linear'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=30, epochs=50, verbose=1)

print(model.summary())

#model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
#model.fit(X, y, epochs=50, batch_size=30)

_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100))
```