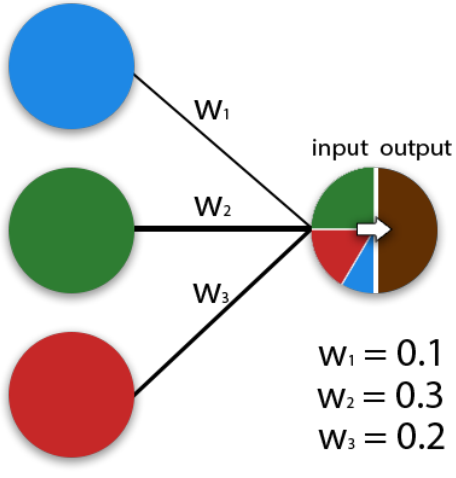
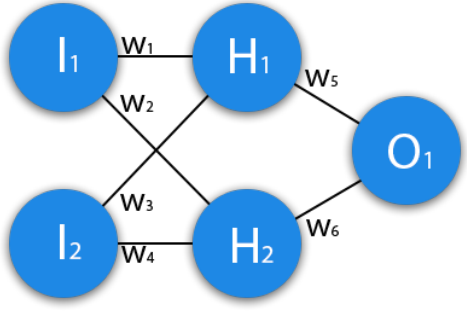
Машинное обучение

Модель нейронной сети

Синапс нейронной сети



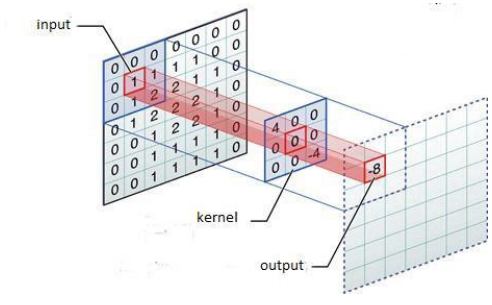
Нейронная сеть со скрытыми слоями



$$H = f_1(W_1 * I)$$
$$O = f_2(W_2 * H)$$

W_1 и W_2 - матрицы весов, обучаемые параметры
I - вектор входных признаков
O - вектор выхода
H - вектор скрытого состояния
f - функция активации

Свёртка



$$O_{i,j} = f(K * I_{i-s:i+s,j-s:j+s})$$

K - ядро свёртки

I - входной тезнор

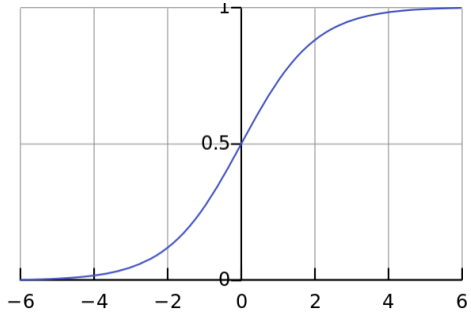
$I_{i-s:i+s,j-s:j+s}$ - срез тензора размера ядра свёртки с центром в точке (i, j)

$O_{i,j}$ - результат свёртки (значение выходного тензора в точке (i, j))

f - функция активации

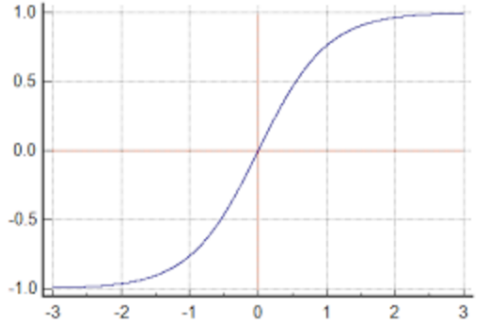
Функция активации

Сигмоида



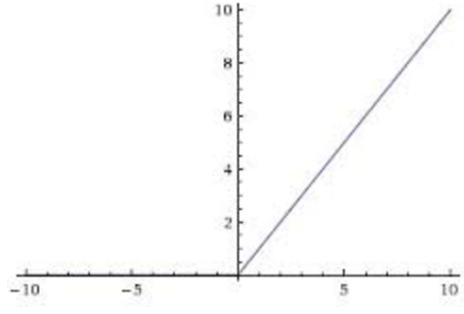
$$F(x) = \frac{1}{1 + e^{-x}}$$

Гиперболический тангенс



$$F(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

ReLU (rectified linear unit)



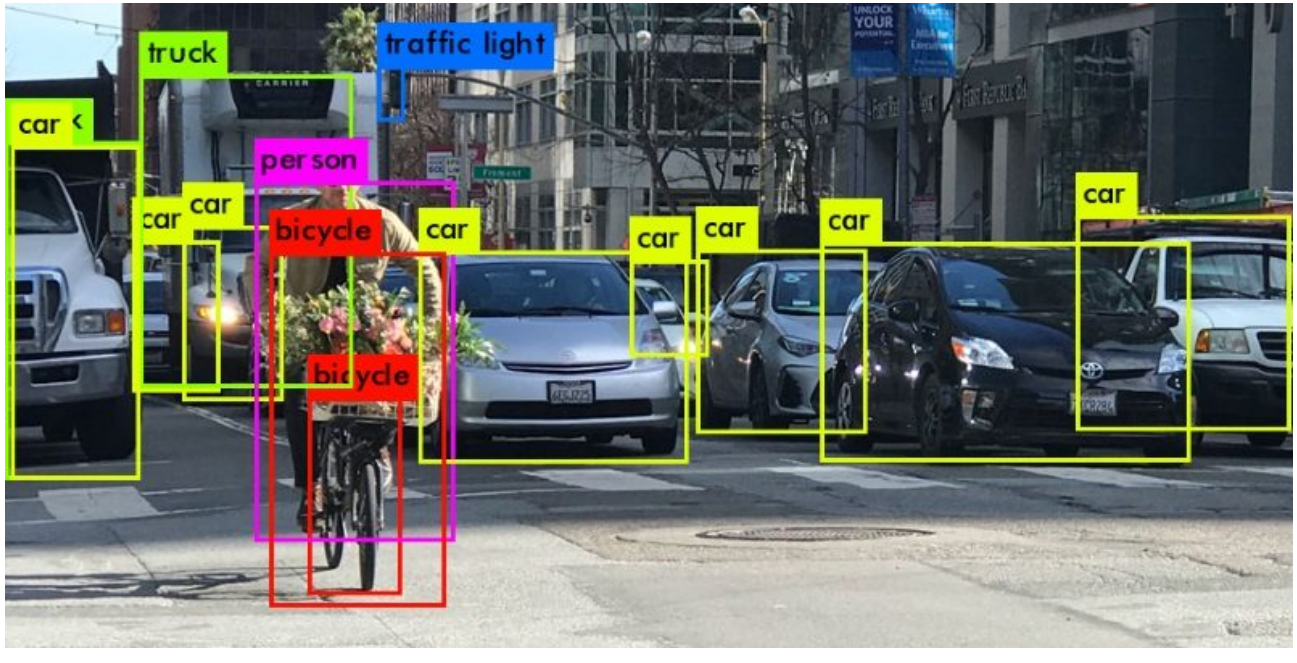
$$F(x) = max(0, x)$$

Разметка данных

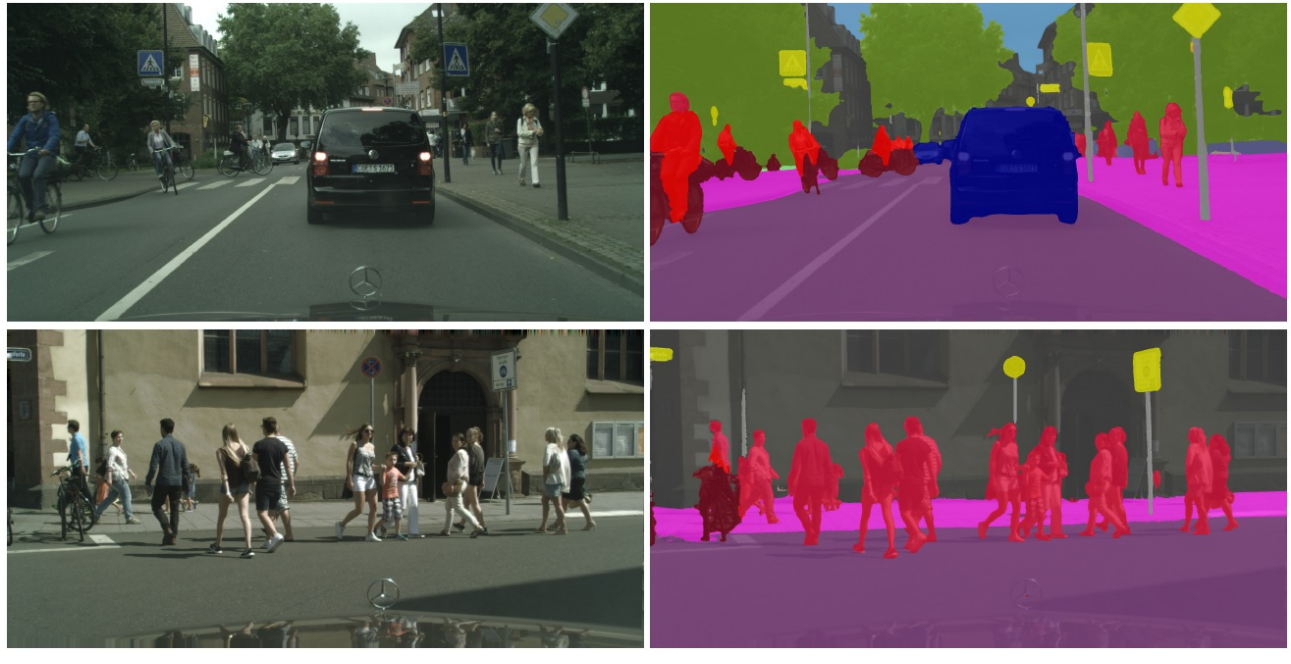
Несмотря на то, что разметка, казалось бы, тривиальная операция – внесение в изображение или текст тэгов, в этих словах содержится глубокий смысл. В процессе разметки производится качественное преобразование - сырые данные дополняются метаданными и превращаются в информацию. Самое утилитарное определение информации звучит следующим образом «Информация – это данные плюс метаданные»

	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 31	7.925		S
5	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
10	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333		S
11	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708		C
12	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549	16.7	G6	S
13	12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.55	C103	S
14	13	0	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2151	8.05		S
15	14	0	3	Andersson, Mr. Anders Johan	male	39	1	5	347082	31.275		S
16	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14	0	0	350406	7.8542		S
17	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55	0	0	248706	16		S
18	17	0	3	Rice, Master. Eugene	male	2	4	1	382652	29.125		Q
19	18	1	2	Williams, Mr. Charles Eugene	male		0	0	244373	13		S
20	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)	female	31	1	0	345763	18		S
21	20	1	3	Masselmani, Mrs. Fatima	female		0	0	2649	7.225		C
22	21	0	2	Fynney, Mr. Joseph J	male	35	0	0	239865	26		S
23	22	1	2	Beesley, Mr. Lawrence	male	34	0	0	248698	13	D56	S
24	23	1	3	McGowan, Miss. Anna "Annie"	female	15	0	0	330923	8.0292		Q
25	24	1	1	Sloper, Mr. William Thompson	male	28	0	0	113788	35.5	A6	S
26	25	0	3	Palsson, Miss. Torborg Danira	female	8	3	1	349909	21.075		S
27	26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia Johansson)	female	38	1	5	347077	31.3875		S
28	27	0	3	Emir, Mr. Farred Chehab	male		0	0	2631	7.225		C

Детектирование



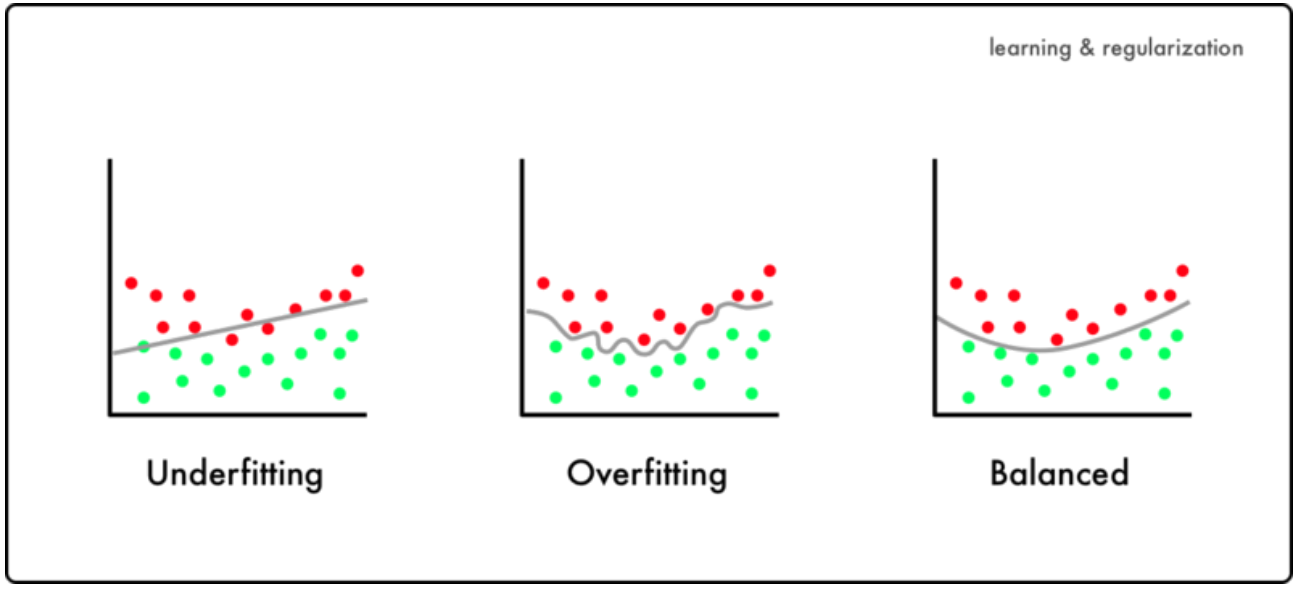
Сегментация



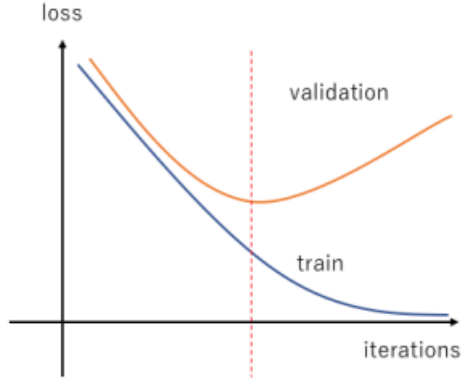
Переобучение

Недообучение — нежелательное явление, возникающее при решении задач обучения по прецедентам, когда алгоритм обучения не обеспечивает достаточно малой величины средней ошибки на обучающей выборке. Недообучение возникает при использовании недостаточно сложных моделей.

Переобучение (overtraining, overfitting) — нежелательное явление, возникающее при решении задач обучения по прецедентам, когда вероятность ошибки обученного алгоритма на объектах тестовой выборки оказывается существенно выше, чем средняя ошибка на обучающей выборке. Переобучение возникает при использовании избыточно сложных моделей.



Процесс обучения



Возможные решения при переобучении:

- Увеличение количества данных в наборе;
- Уменьшение количества параметров модели (количество параметров модели (весов) была в 2 - 3 раза меньше числа примеров обучающего множества);
- Добавление регуляризации / увеличение коэффициента регуляризации.

Возможные решения при недообучении:

- Добавление новых параметров модели;
- Использование для описания модели функций с более высокой степенью;
- Уменьшение коэффициента регуляризации.

Аугментация данных

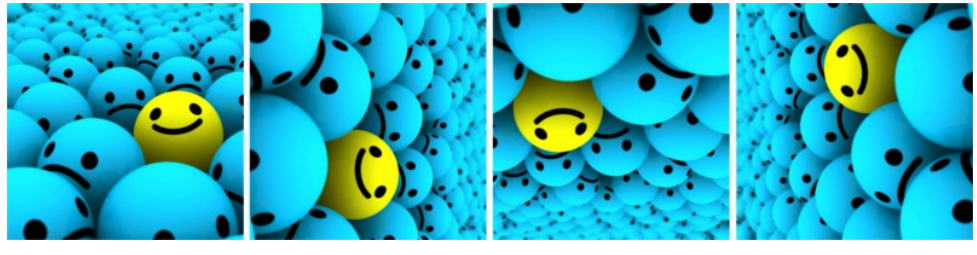
Аугментация данных (data augmentation) – это методика создания дополнительных обучающих данных из имеющихся данных. Для достижения хороших результатов глубокие сети должны обучаться на очень большом объеме данных. Следовательно, если исходный обучающий набор содержит ограниченное количество изображений, необходимо выполнить аугментацию, чтобы улучшить результаты модели.

Можно использовать следующие искажения:

- Геометрические (афинные, проективные, ...);
- Яркостные/цветовые;
- Замена фона;
- Искажения, характерные для решаемой задачи: блики, шумы, размытие и т. д.

Отражение

Поворот

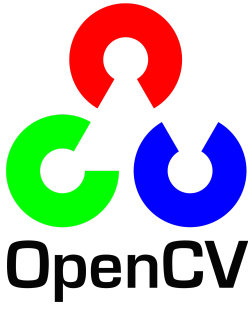


Случайное образание

Шумы

Аугментация с помощью GAN (Generative adversarial network - Генеративно-состязательная сеть)

Библиотека OpenCV

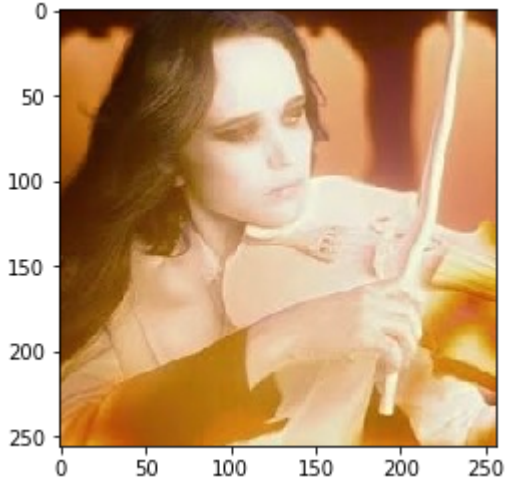


OpenCV (Open Source Computer Vision Library) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков

In [11]:

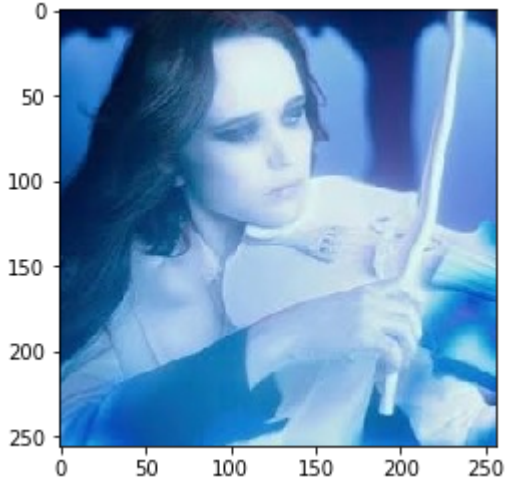
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
# Чтение картинки (чтение происходит в цветовой модели BGR)
# img_bgr = cv2.imread('data/Lenna.png')
img_bgr = cv2.imread('data/Vanya.jpg')
print(img_bgr.shape)
plt.imshow(img_bgr)
plt.show()
```

```
(256, 256, 3)
```



In [2]:

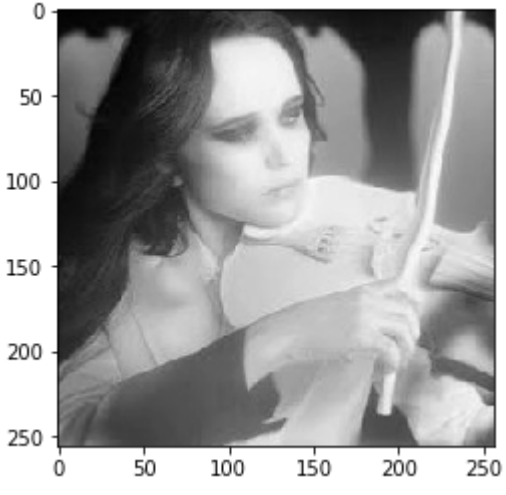
```
# Для перевода в RGB можно либо воспользоваться функцией opencv, либо инвертировать каналы
img = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
img = img_bgr[:, :, ::-1]  # Каналы - третье измерение изображения
plt.imshow(img)
plt.show()
```



In [3]:

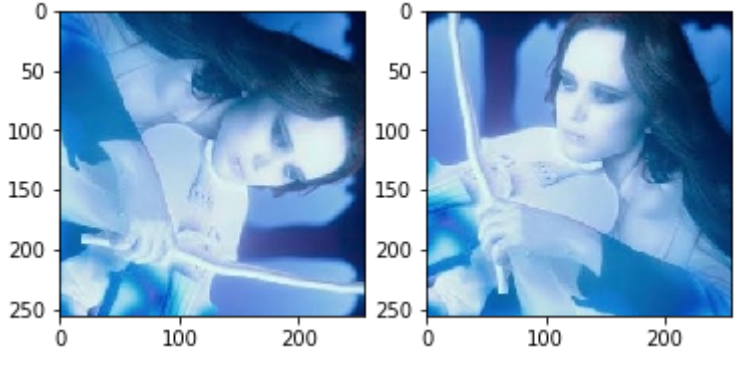
```
# Есть возможность сразу читать в цветовой модели Grayscale
img_gray = cv2.imread('data/Vanya.jpg', 0)
print(img_gray.shape)
plt.imshow(img_gray, cmap='gray')
plt.show()
```

```
(256, 256)
```



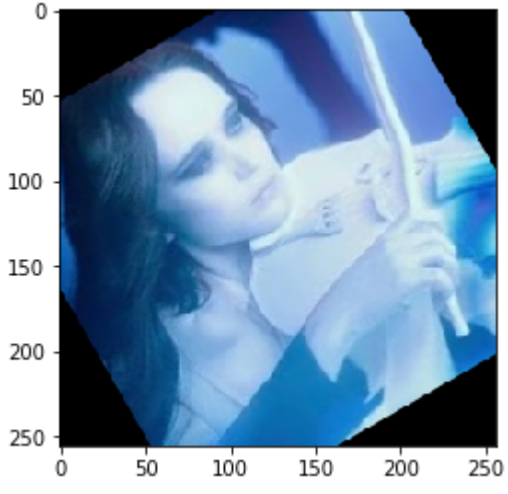
Преобразования изображений

In [4]:
```
# Поворот изображения на 90 градусов
img_rot = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
# Отражение по горизонтали (0 - по вертикали)
img_flip = cv2.flip(img, 1)
plt.subplot(1, 2, 1)
plt.imshow(img_rot)
plt.subplot(1, 2, 2)
plt.imshow(img_flip)
plt.show()
```



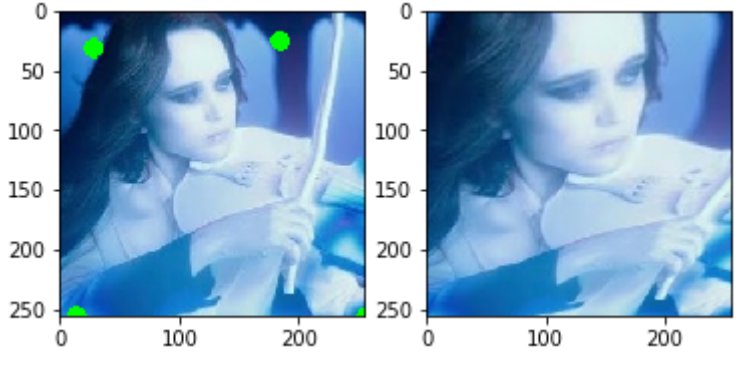
In [5]:

```
# Поворот на произвольный угол с помощью афинных преобразований
M = cv2.getRotationMatrix2D((128, 128), 30, 1)  # Указывается центр, угол поворота, масштаб
img_rot = cv2.warpAffine(img, M, (256, 256))
plt.imshow(img_rot)
plt.show()
```



In [6]:

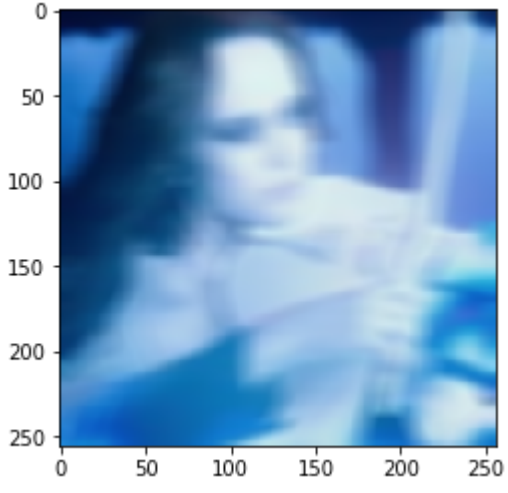
```
# Изменение перспективы
# pts1 = np.float32([[56, 65],[368, 52],[28, 512],[412, 512]])
# pts2 = np.float32([[0, 0], [512, 0], [0, 512], [512, 512]])
pts1 = np.float32([[ 28. ,  32.5],
                   [184. ,  26. ],
                   [ 14. , 256. ],
                   [256. , 256. ]])
pts2 = np.float32([[0, 0], [256, 0], [0, 256], [256, 256]])

M = cv2.getPerspectiveTransform(pts1,pts2)

img_persp = cv2.warpPerspective(img, M, (256, 256))

# Нарисуем исходные точки (изменяет исходное изображение)
img_copy = img.copy()
for pt in pts1:
    cv2.circle(img_copy, tuple(pt), 4, (0, 255, 0), 8)

plt.subplot(1, 2, 1)
plt.imshow(img_copy)
plt.subplot(1, 2, 2)
plt.imshow(img_persp)
plt.show()
```



In [7]:

```
# Изменение размера
img_sc = cv2.resize(img, (512, 256))
plt.imshow(img_sc)
plt.show()
```

In [8]:
```
# Размытие
img_bl = cv2.blur(img, (17, 5))
plt.imshow(img_bl)
plt.show()
```



Генераторы в Python

Генераторы и итераторы представляют собой инструменты, которые, как правило, используются для поточной обработки данных.

Итератор представляет собой объект перечислитель, который для данного объекта выдает следующий элемент, либо бросает исключение, если элементов больше нет.

Создание итератора

In [9]:

```
# Из списка
it = iter([1, 2, 5, 3])
print(it)
# Перечисление циклом for
for i in it:
    print(i)
```

```
<list_iterator object at 0x7fcc5827a048>
1
2
5
3
```

In [10]:

```
# Перечисление с помощью функции next
it = iter([1, 2, 5, 3])
while True:
    print(next(it))
```

```
1
2
5
3
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-10-fb8c1e96a5f8> in <module>()
      2 it = iter([1, 2, 5, 3])
      3 while True:
----> 4     print(next(it))

StopIteration:
```

In [12]:

```
# Создание в функции
def fun_iter(val):
    s = 1
    while True:
        s = s * val
        yield s

fit = fun_iter(2)
for i in range(4):
    print(next(fit))
```

```
2
4
8
16
```

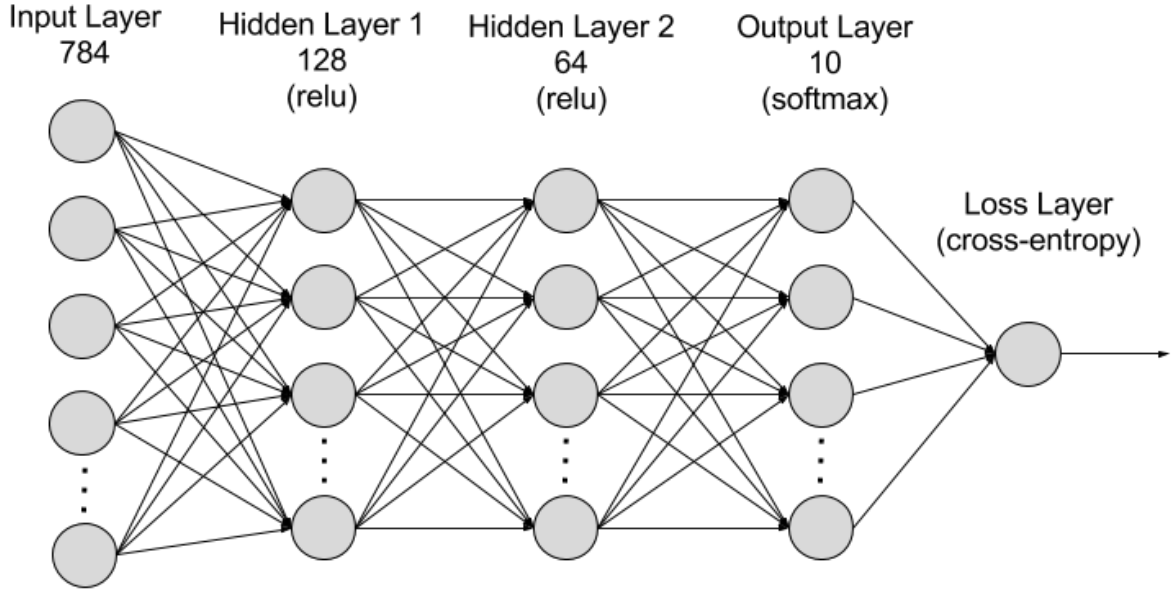
Softmax слой

Softmax — это обобщение логистической функции для многомерного случая. Функция преобразует вектор z размерности K в вектор σ той же размерности, где каждая координата σ_i полученного вектора представлена вещественным числом в интервале [0,1] и сумма координат равна 1.

$$\sigma_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

Функция Softmax применяется в машинном обучении для задач классификации, когда количество возможных классов больше двух (для двух классов используется логистическая функция). Координаты σ_i полученного вектора при этом трактуются как вероятности того, что объект принадлежит к классу i.
Часто Softmax используется для последнего слоя глубоких нейронных сетей для задач классификации.

Нейронная сеть с softmax слоем

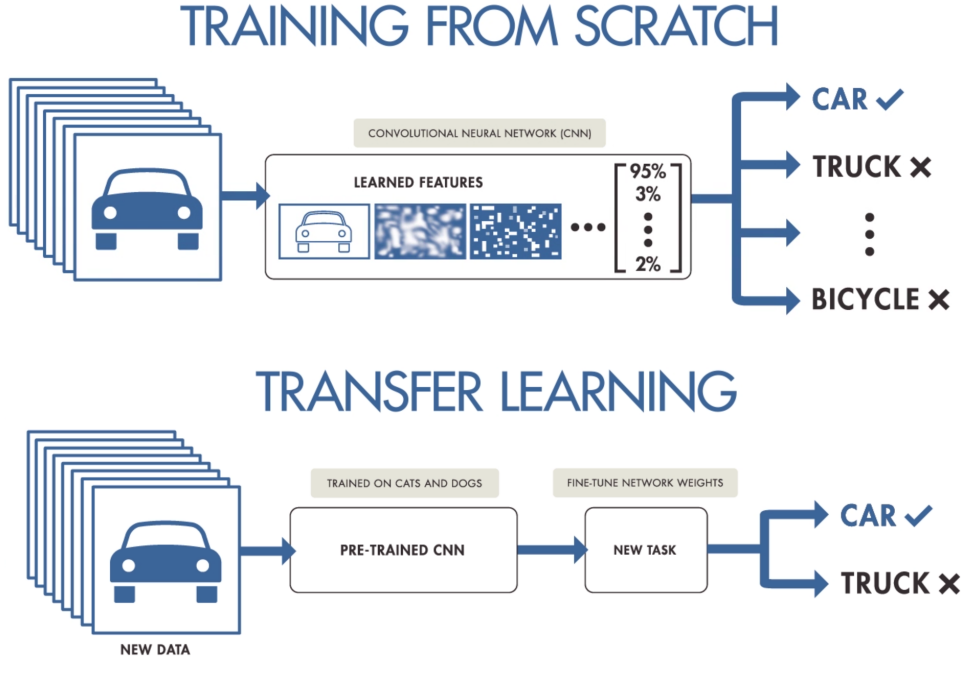


Output Layer возвращает вероятности принадлежности примера к каждому из 100 классов

Hidden Layer 2 возвращает вектор скрытых признаков

Дообучение

Использование уже предобученных сетей на подобных задачах позволяет получить хороший результат с использованием меньшего количества времени и данных



Задания

1. Скачать датасет Nails segmentation (https://www.kaggle.com/vpapenko/nails-segmentation#1eecab90-1a92-43a7-b952-0204384e1fae.jpg). Составить список из пар (<имя изображения>, <маска>) для всех данных, используя функцию os.listdir() или glob.glob().
2. Создать генератор, который на каждой итерации возвращает пару списков из заданного количества (аргумент функции) изображений и масок к ним (итератор должен перемешивать примеры).
3. Добавить в генератор случайную аугментацию (каждая применяется случайно). После преобразований все изображения должны иметь одинаковый размер. *Обратите внимание, что большинство преобразований должны применяться одинаково к изображению и маске*
    A. Поворот на случайный угол
    B. Отражение по вертикали, горизонтали
    C. Вырезание части изображения
    D. Размытие