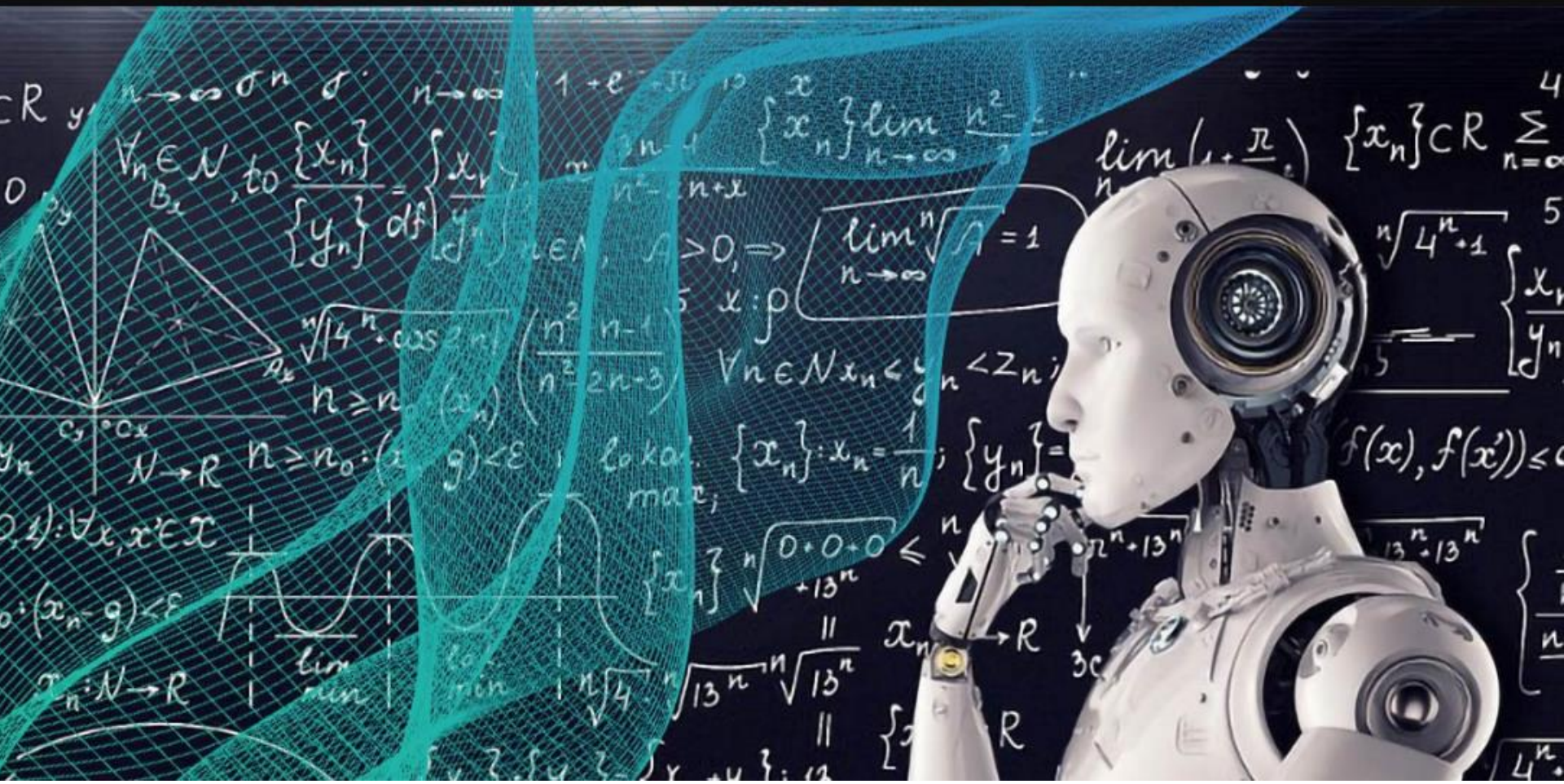


ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

И компьютерное зрение

НЕМНОГО ИСТОРИИ

Первые идеи – 1950-1960 годы

Период разочарования – до 1990-х

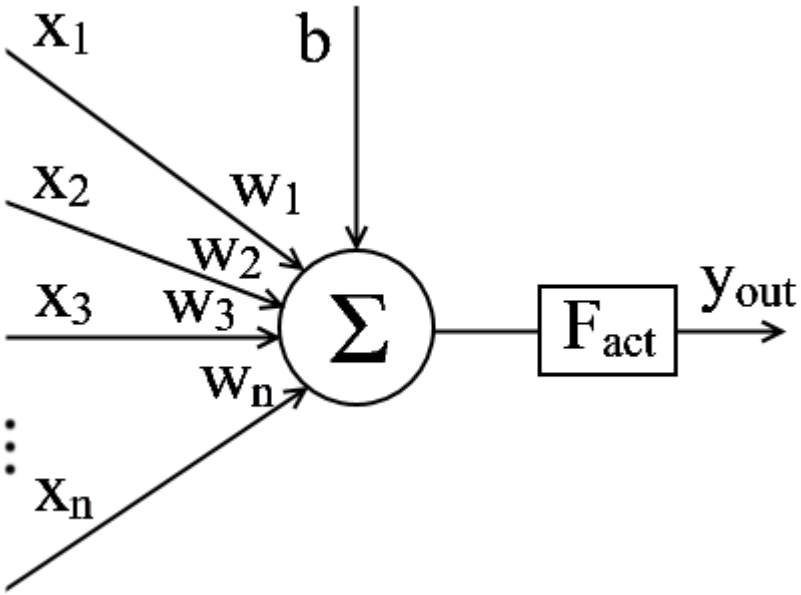
С 2000-х началось активное развитие

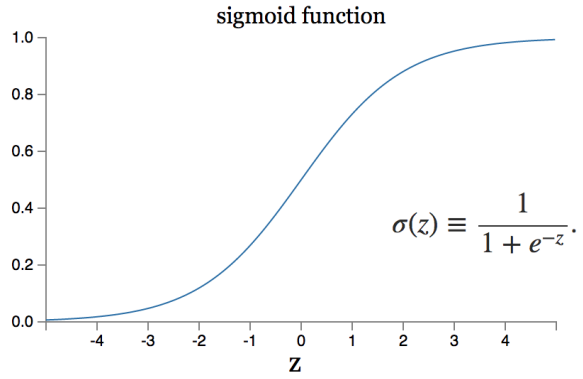
В 2012 появились глубокие нейросети

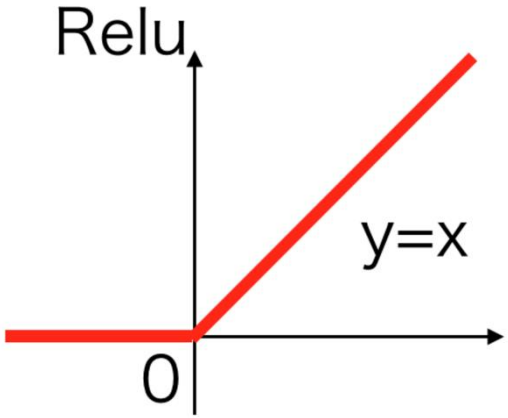
В 2015 на датасете ImageNet был превзойден человеческий уровень распознавания

ПЕРСЕПТРОН (НЕЙРОН)

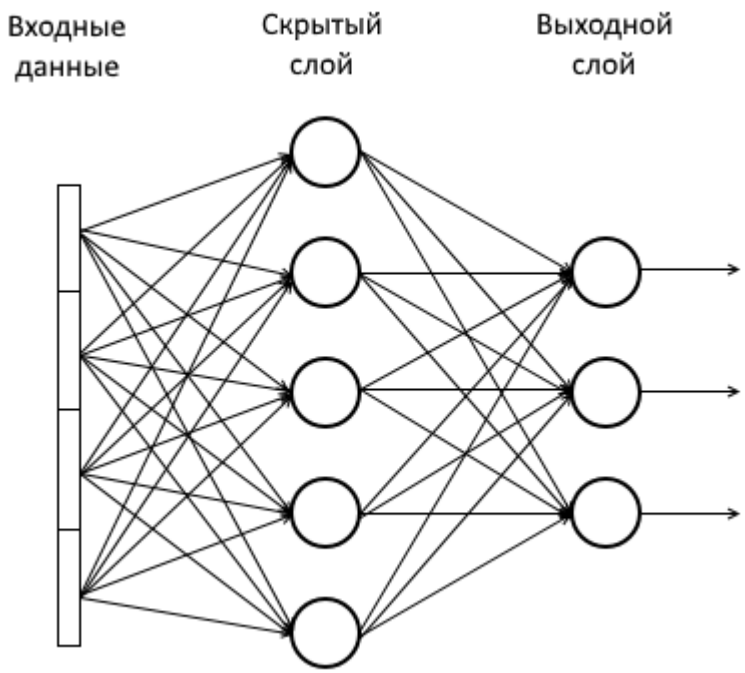
$$y_{out} = F_{act}(b + \sum_{i=1}^{n}(w_i * x_i))$$



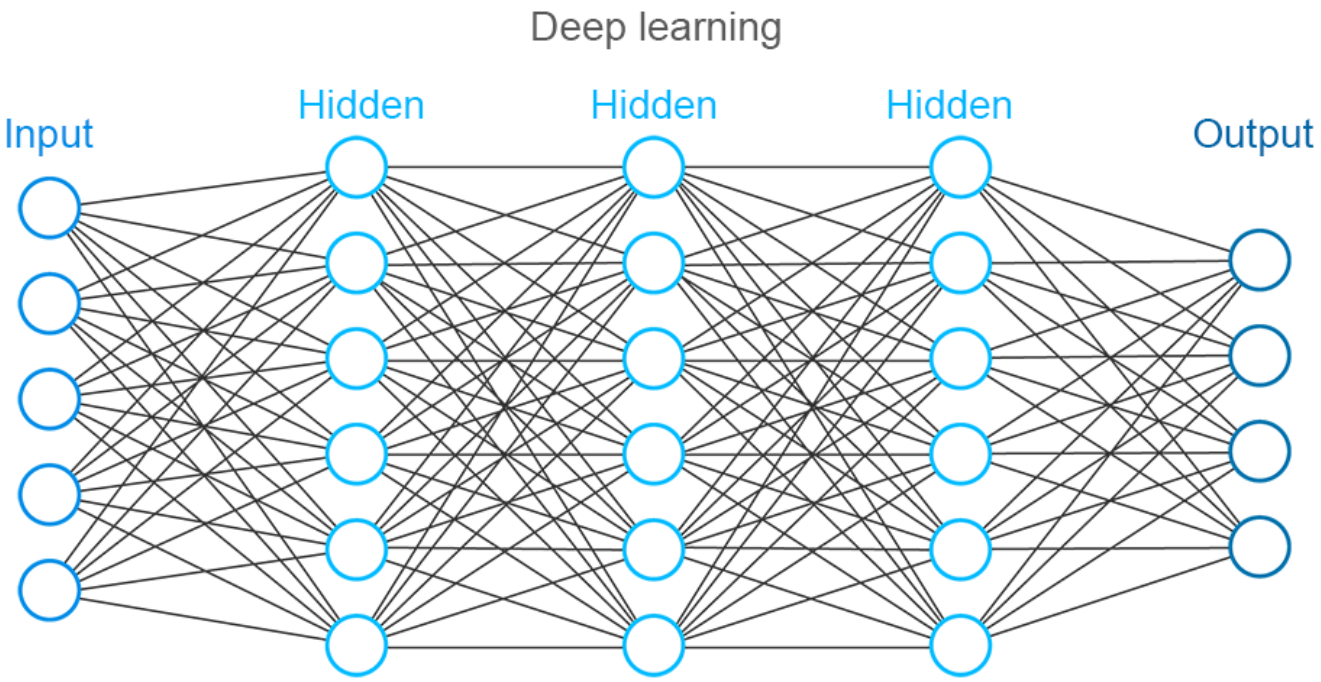




НЕЙРОСЕТЬ



ГЛУБОКАЯ НЕЙРОСЕТЬ



ВИДЫ НЕЙРОСЕТЕЙ

1) Сверточные

2) Сеть Кохонена, Хопфилда

3) Рекуррентные

4) Спайковые

И т.д.

ПОЧЕМУ НЕЙРОСЕТИ СЕЙЧАС ВЕЗДЕ?

1) Железо

2) Данные

3) Модно

МАШИННОЕ ОБУЧЕНИЕ

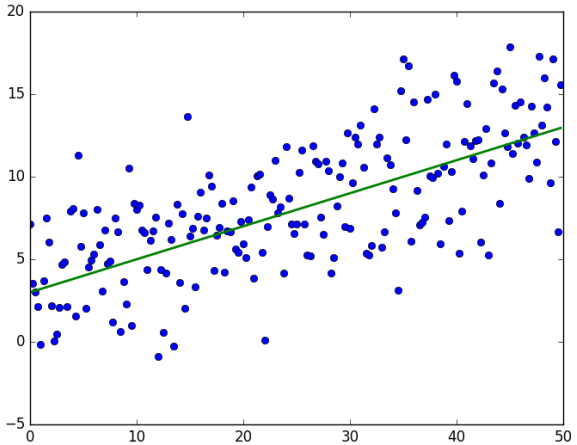
Методы и алгоритмы, позволяющие выполнять задачу без четкого описания решения.

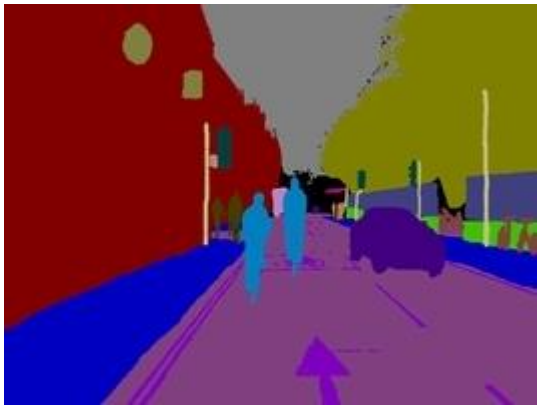
1) Обучение на прецедентах

2) Минимизация ошибки

3) Метрика качества

ОБУЧЕНИЕ НА ПРИМЕРАХ





Признаки (X)					Целевая переменная (Y)	
Рост	**IQ**	**Пиво в день**	**Пол**	**Ср. бал в школе**	**Ср. бал в универ.**	**Курит?**
170	128	0	М	4.7	4.4	Нет
195	90	40	М	3.3	3.1	Да
160	111	2	Ж	4.0	3.9	Нет
183	143	0	М	4.8	4.7	Да

ЗАДАЧИ МАШИННОГО ОБУЧЕНИЯ

1) Классификация

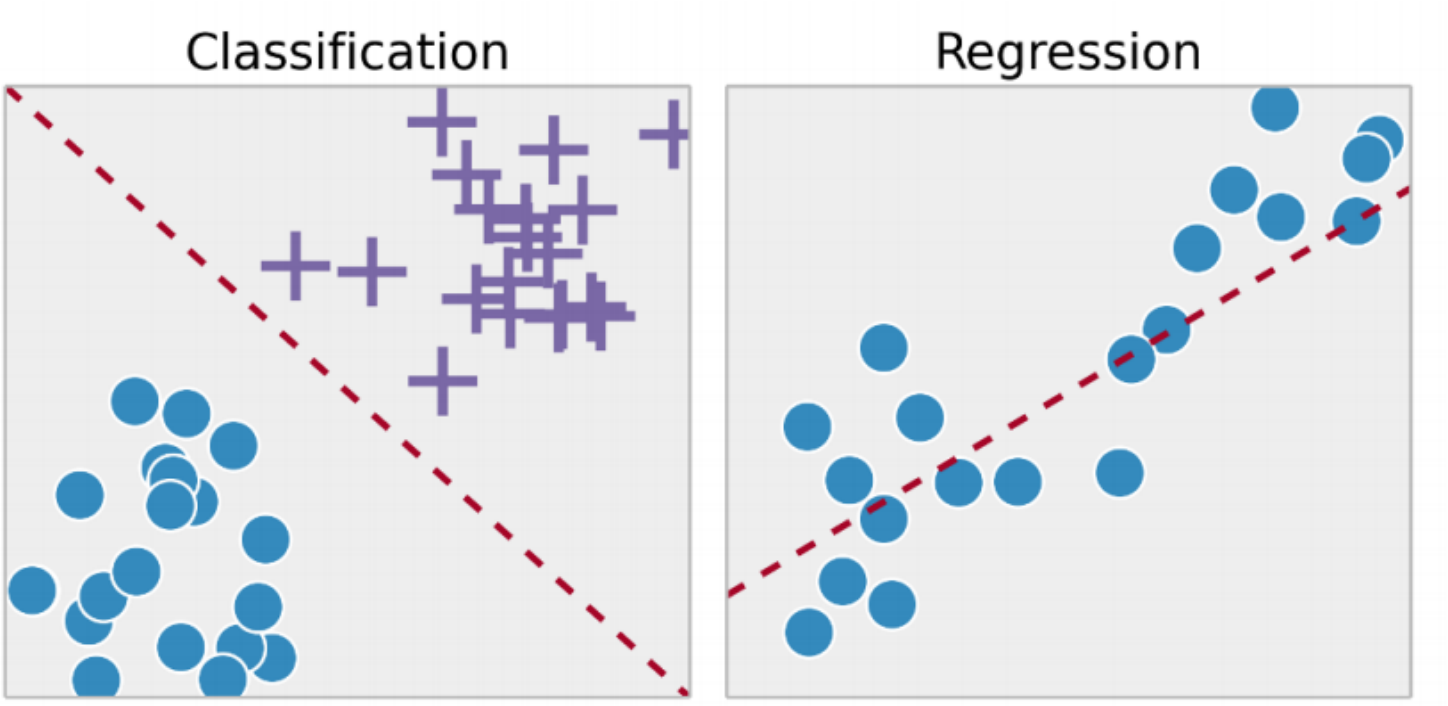
2) Регрессия

3) Кластеризация

4) Ранжирование

5) Предсказание

КЛАССИФИКАЦИЯ И РЕГРЕССИЯ



МИНИМИЗАЦИЯ ОШИБКИ

Подбор таких параметров модели, при которых ошибка (loss) будет минимальной.

Функции потерь для задач классификации:

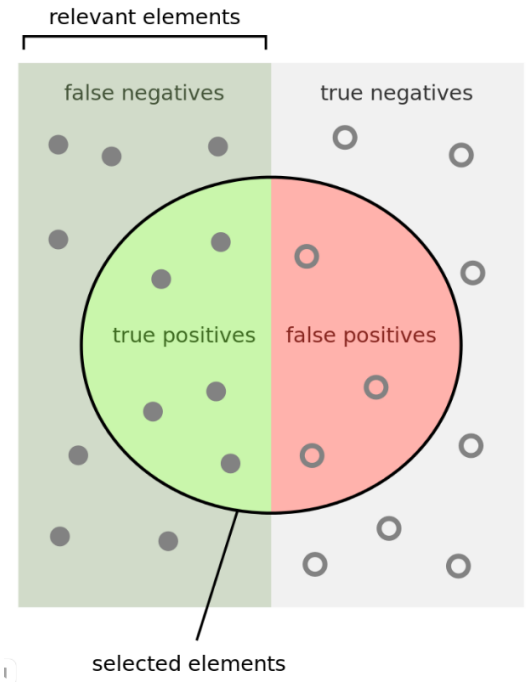
- $\mathscr{L}(a, x) = [a(x) \neq y(x)]$ — индикатор ошибки;

Функции потерь для задач регрессии:

- $\mathscr{L}(a, x) = (a(x) - y(x))^2$ — квадратичная ошибка.

МЕТРИКА КАЧЕСТВА

Accuracy – доля верных ответов во всех ответах. Понятно, но иногда бесполезно



How many selected items are relevant?

Precision =

How many relevant items are selected?

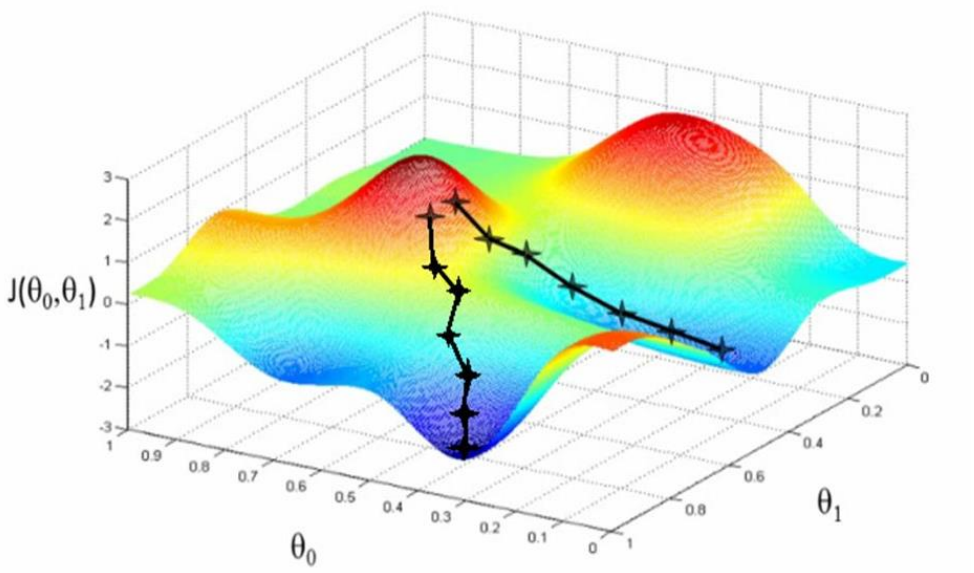
Recall =

ОБУЧЕНИЕ МОДЕЛИ

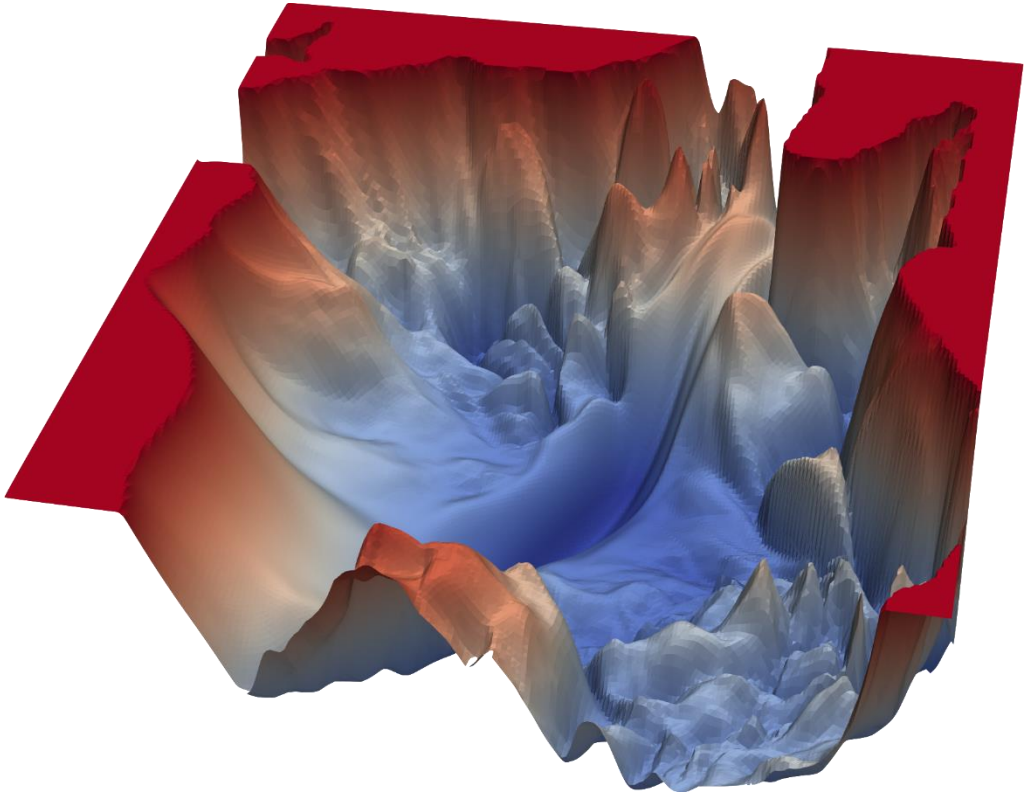
Обучение = уменьшение ошибки

Нейронная сеть – модель с безумным числом параметров – от нескольких сотен до десятков миллионов

Градиентный спуск – наше всё



КАК НАЙТИ МИНИМУМ?



ВАРИАЦИИ ГРАДИЕНТНОГО СПУСКА

1) С моментом Нестерова – запоминаем свое прошлое направление движение, «инерция»

2) AdaGrad – учитываем скорость изменения каждого параметра

3) AdaDelta – то же, что AdaGrad, но с нормализацией, что уменьшает эффект паралича сети

4) Adam – практически смесь всех предыдущих

КАК НАЙТИ ГРАДИЕНТ?

Фреймворки решают эту задачу за нас. Но выбор и количество данных при подсчете – наша задача.

1) Стандартный (пакетный) режим – по всем имеющимся данным

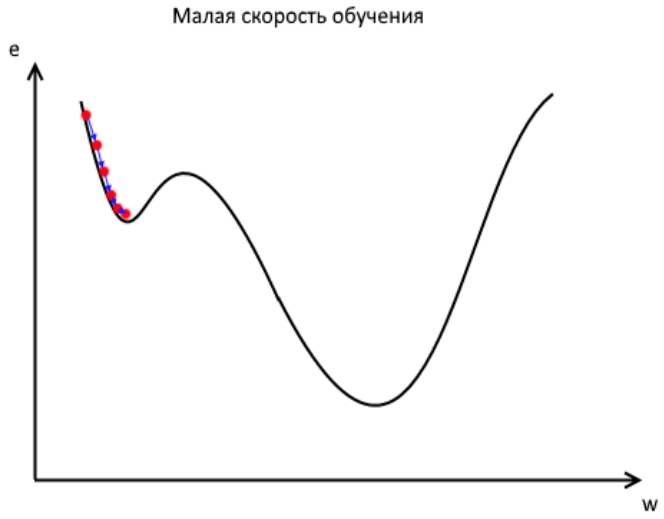
2) Стохастический – по 1 примеру за раз

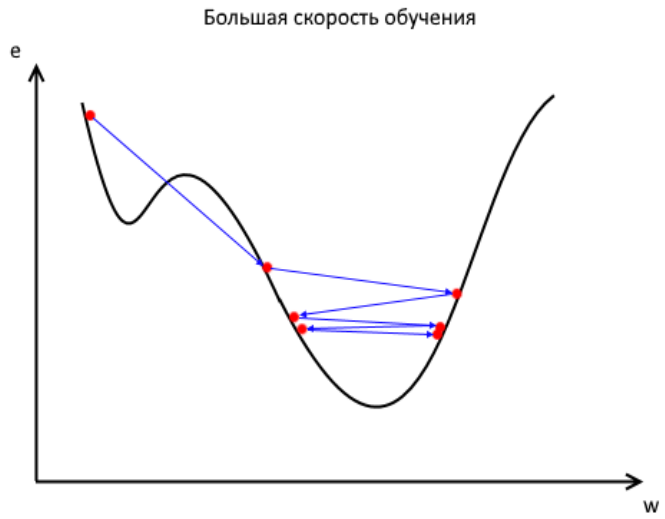
3) Мини-пакетный, mini−batch – по небольшой части тренировочных данных, обычно число кратное 2

ГРАДИЕНТ НАШЛИ, ЧТО ДАЛЬШЕ?

Его нужно применить с определенным коэффициентом – learning rate, скорость обучения, величина шага.

Типичные значения для него – от 0.0001 до 0.1, в зависимости от сети.





КАК ОШИБКА ВЛИЯЕТ НА ПАРАМЕТРЫ МОДЕЛИ?

* - обратное распространение ошибки

РАЗДЕЛЕНИЕ ДАННЫХ

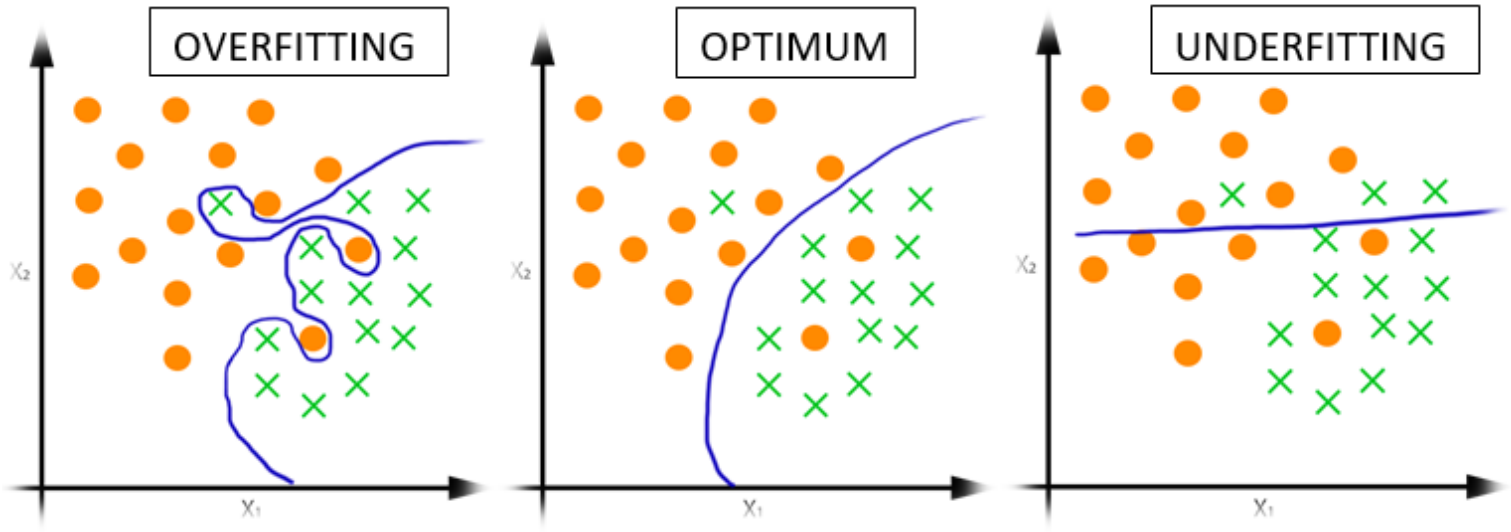
1) Обучающая выборка – на ней обучаемся

2) Валидационная выборка – на ней проверяем качество сети

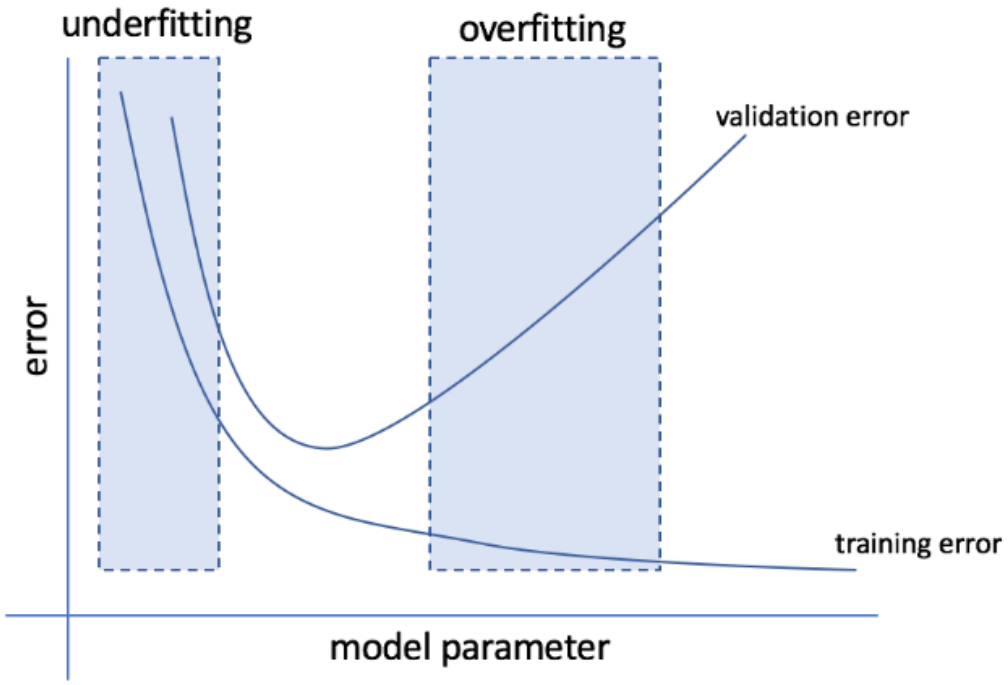
3) Тестовая выборка – не трогаем ни разу, до тех пор, пока все не закончим. Это данные из «реального мира»

Эпоха – проход по всем данным.

УЧИТЬСЯ, УЧИТЬСЯ И ПЕРЕОБУЧИТЬСЯ



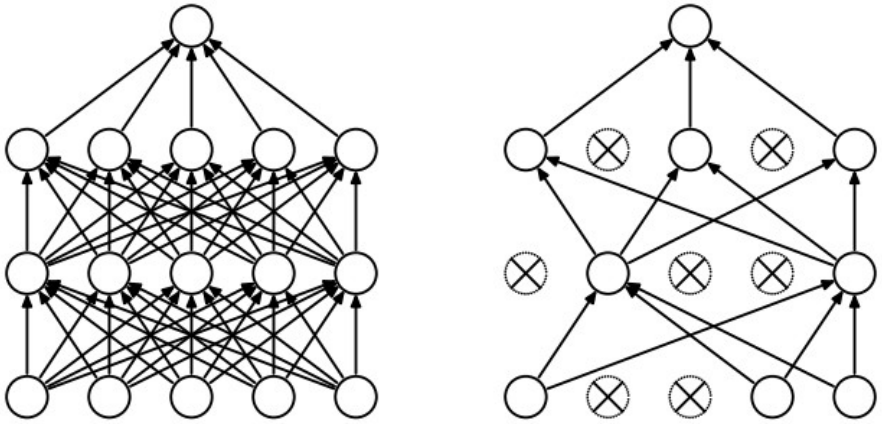
КАК ЭТО ВЫГЛЯДИТ



КАК С ЭТИМ БОРОТЬСЯ

Наивный подход – ограничить нейросеть в умственных способностях (неверно и негуманно).

Правильный подход – использовать L1 и L2 регуляризацию, а так же dropout или batch normalization.



НЕЙРОСЕТЬ

1) Сверточные слои

2) Слои подвыборки (пулинга)

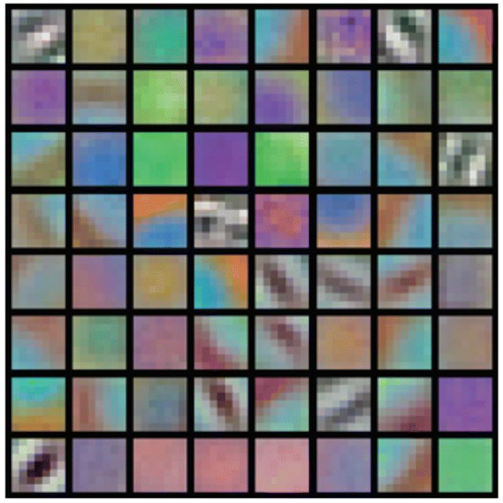
3) Полносвязные слои

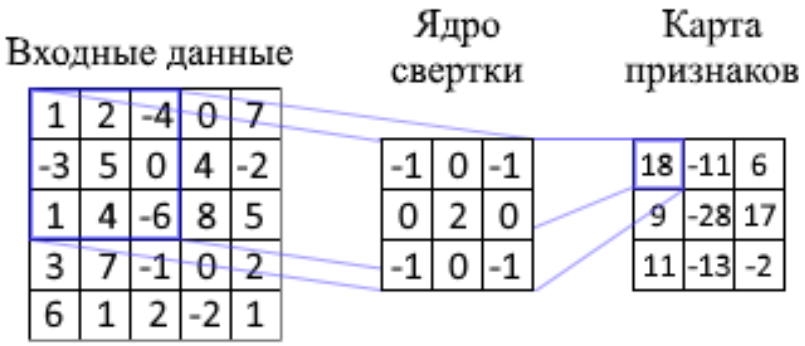
СВЕРТОЧНОЙ СЛОЙ

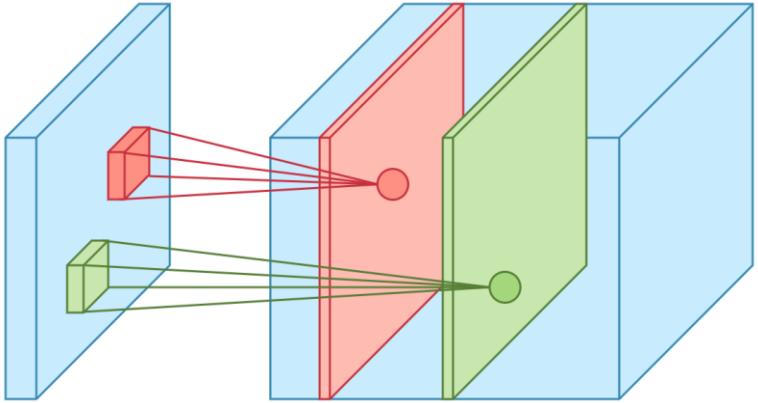
1) Размер ядра (3, 5)

2) Сохранение размера

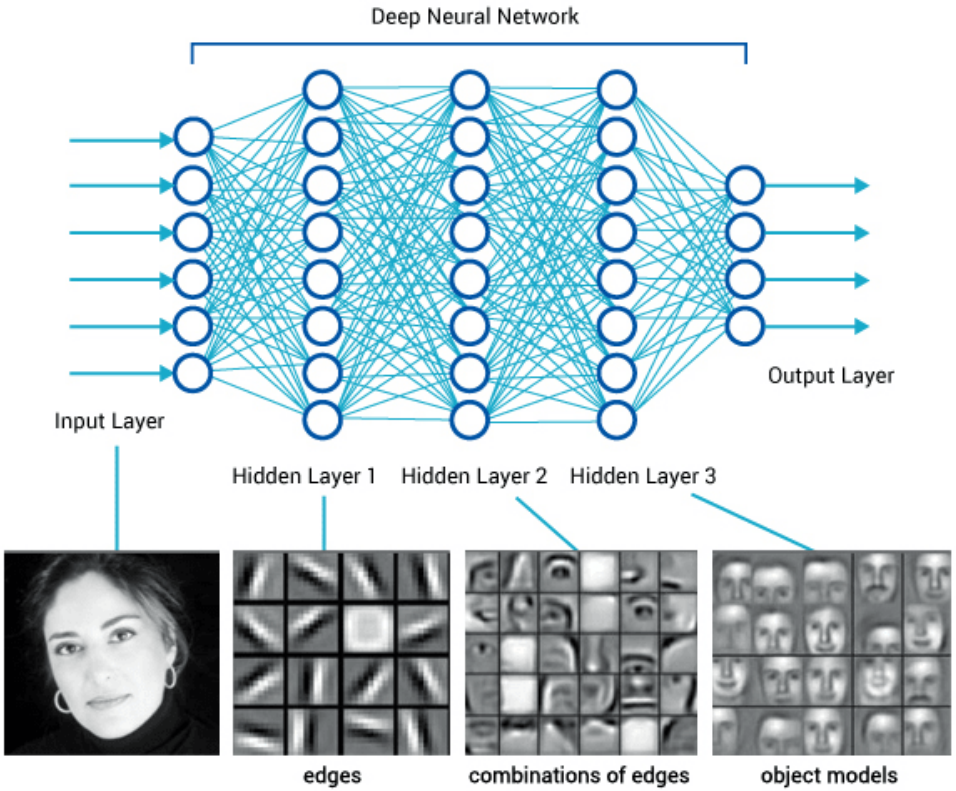
3) Шаг







ПРИМЕР ВЫУЧЕННЫХ ПРИЗНАКОВ

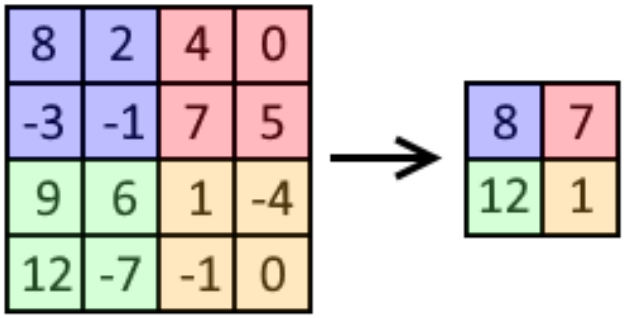


СЛОЙ ПУЛИНГА

1) Размер (2 – 4)

2) Шаг (кратен размеру)

3) Тип (по максимальному)



Уменьшение расчетов за счет потери информации!

ПОЛНОСВЯЗНЫЙ СЛОЙ

1) Ставятся в конце

2) Количество нейронов в последнем слое равно количеству классов (кроме бинарного случая)

ПРАКТИЧЕСКАЯ ЧАСТЬ

1) Python

2) TensorFlow, Keras

3) Google Colab

PYTHON

Python – интерпретируемый язык с динамической типизацией.

Jupyter Notebook – «IDE» в браузере, которая позволяет поэтапно выполнять код.

Удобство и скорость прототипирования привело к тому, что все популярные фреймворки имеют питоновский интерфейс, а иногда и только его.

TENSORFLOW

TF – фреймворк для разработки нейросетей от гугла. В нем есть все современные слои, есть интерфейсы как на питоне, так и на C++.

Keras – надстройка над TF, еще сильнее упрощает и ускоряет разработку нейросетей.

Есть и аналоги – PyTorch, Caffe, MXNet, DL4J, …

GOOGLE COLAB

Бесплатный облачный jupyter notebook с GPU!

Можно сохранять свои тетрадки в гугл диске.

Есть ограничения на время работы, а так же могут быть проблемы с загрузкой датасетов.

ПРАКТИЧЕСКАЯ ПРАКТИКА

Задача – создать свою нейросеть и обучить на датасете MNIST – «hello, world!» в мире нейросетей.

ПОДКЛЮЧЕНИЕ БИБЛИОТЕК

```
[3]  from keras.datasets import mnist
     from keras.models import Sequential
     from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout
     from keras.optimizers import Adam
     from keras.utils import to_categorical
     import keras
     import keras.backend as K

     import matplotlib.pyplot as plt
     import numpy as np
```

```
Using TensorFlow backend.
```

ОБЩИЕ ПЕРЕМЕННЫЕ

```
[4]  batch_size = 256
     num_classes = 10
     epochs = 10
```

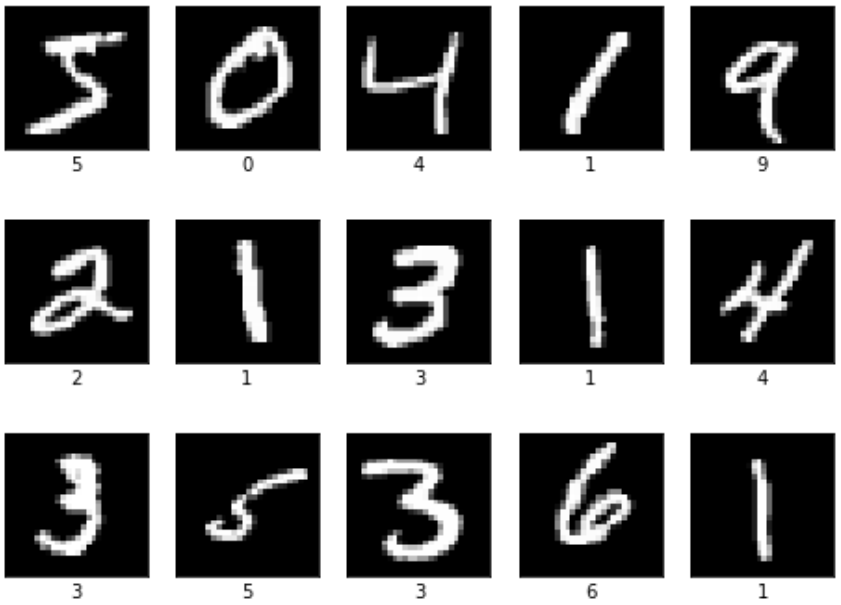
ЗАГРУЗКА ДАТАСЕТА

```
[5]  (x_train, y_train), (x_test, y_test) = mnist.load_data()

     img_rows = 28
     img_cols = 28

     if K.image_data_format() == 'channels_first':
         x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
         x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
         input_shape = (1, img_rows, img_cols)
     else:
         x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
         x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
         input_shape = (img_rows, img_cols, 1)

     print('data train shape: {0}, label train shape: {1}'.format(x_train.shape, y_train.shape))
     print('data test shape: {0}, label test shape: {1}'.format(x_test.shape, y_test.shape))
```

```
data train shape: (60000, 28, 28, 1), label train shape: (60000,)
data test shape: (10000, 28, 28, 1), label test shape: (10000,)
```

ЧТО ЗА ДАННЫЕ?

```
[6]  fig = plt.figure(figsize=(8, 6))

     for i in range(15):
         ax = fig.add_subplot(3, 5, i + 1, xticks=[], yticks=[])
         ax.set_xlabel(y_train[i])
         ax.imshow(x_train[i,:,:,0], cmap='gray')
```



ПРЕДОБРАБОТКА ДАННЫХ

```
[7]  x_train = x_train.astype('float32')
     x_test = x_test.astype('float32')
     x_train /= 255.0
     x_test /= 255.0
     print(x_train.min(), x_train.max())
     print(x_test.min(), x_test.max())
```

```
0.0 1.0
0.0 1.0
```

```
[8]  y_train = to_categorical(y_train, num_classes)
     y_test = to_categorical(y_test, num_classes)
     print(y_train[0])
```

```
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

АРХИТЕКТУРА СЕТИ

```
[31]  model = Sequential()
      model.add(Conv2D(32, kernel_size=(3, 3),
                       activation='relu',
                       input_shape=input_shape))
      model.add(MaxPooling2D(pool_size=(2, 2)))
      model.add(Conv2D(64, (3, 3), activation='relu'))
      model.add(MaxPooling2D(pool_size=(2, 2)))
      model.add(Flatten())
      model.add(Dense(64, activation='relu'))
      model.add(Dropout(0.5))
      model.add(Dense(num_classes, activation='softmax'))
```

ИНФОРМАЦИЯ О СЕТИ

```
[32] model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 5, 5, 64)          0
_________________________________________________________________
flatten_7 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_17 (Dense)             (None, 64)                102464
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_18 (Dense)             (None, 10)                650
=================================================================
Total params: 121,930
Trainable params: 121,930
Non-trainable params: 0
_________________________________________________________________
```

МЕТОД ОБУЧЕНИЯ И МЕТРИКИ

```
[33]  model.compile(loss = keras.losses.categorical_crossentropy,
                    optimizer = Adam(),
                    metrics = ['accuracy'])
```

ЗАПУСК ОБУЧЕНИЯ

```
history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_split=0.3
                    )

score = model.evaluate(x_test, y_test, verbose=0)
print('\n\nTest loss:', score[0])
print('Test accuracy:', score[1])
```

ПРОЦЕСС ОБУЧЕНИЯ

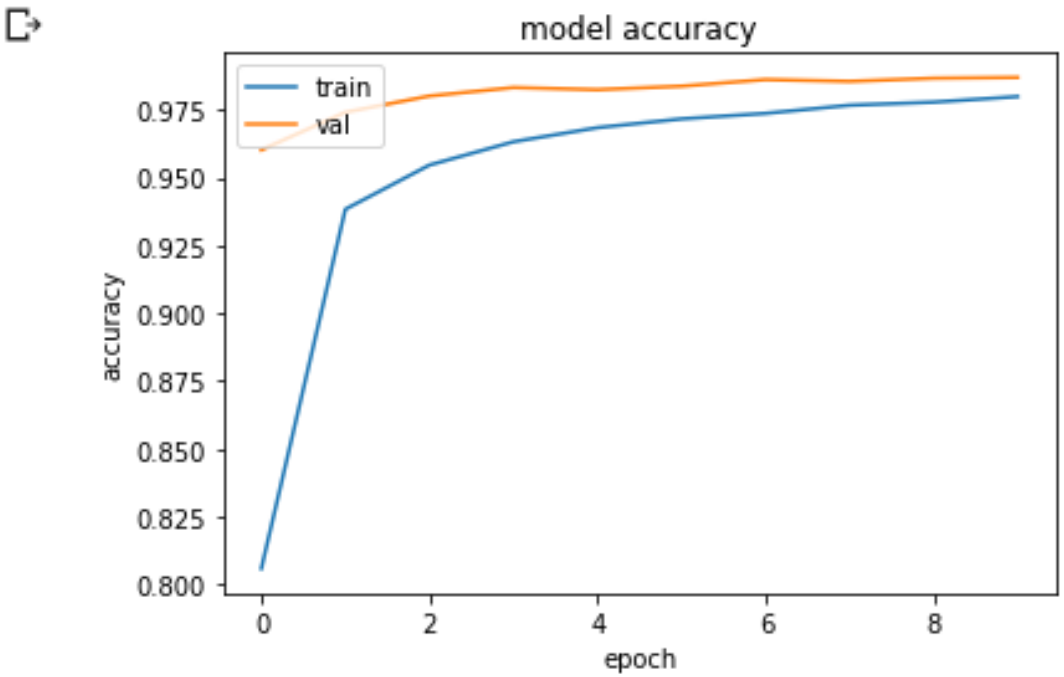
```
Train on 42000 samples, validate on 18000 samples
Epoch 1/10
42000/42000 [==============================] - 34s 800us/step - loss: 0.6248 - acc: 0.8058 - val_loss: 0.1307 - val_acc: 0.9602
Epoch 2/10
42000/42000 [==============================] - 33s 788us/step - loss: 0.2070 - acc: 0.9383 - val_loss: 0.0842 - val_acc: 0.9741
Epoch 3/10
42000/42000 [==============================] - 33s 787us/step - loss: 0.1534 - acc: 0.9546 - val_loss: 0.0699 - val_acc: 0.9801
Epoch 4/10
42000/42000 [==============================] - 33s 786us/step - loss: 0.1250 - acc: 0.9632 - val_loss: 0.0585 - val_acc: 0.9832
Epoch 5/10
42000/42000 [==============================] - 33s 784us/step - loss: 0.1090 - acc: 0.9684 - val_loss: 0.0571 - val_acc: 0.9824
Epoch 6/10
42000/42000 [==============================] - 33s 782us/step - loss: 0.0958 - acc: 0.9716 - val_loss: 0.0562 - val_acc: 0.9837
Epoch 7/10
42000/42000 [==============================] - 33s 785us/step - loss: 0.0878 - acc: 0.9736 - val_loss: 0.0477 - val_acc: 0.9861
Epoch 8/10
42000/42000 [==============================] - 33s 785us/step - loss: 0.0755 - acc: 0.9766 - val_loss: 0.0498 - val_acc: 0.9854
Epoch 9/10
42000/42000 [==============================] - 33s 788us/step - loss: 0.0708 - acc: 0.9778 - val_loss: 0.0477 - val_acc: 0.9866
Epoch 10/10
42000/42000 [==============================] - 33s 785us/step - loss: 0.0660 - acc: 0.9798 - val_loss: 0.0465 - val_acc: 0.9869


Test loss: 0.030736874192990035
Test accuracy: 0.9892
```

ПЕЧАТЬ ГРАФИКА ОБУЧЕНИЯ

```
[22]  def plot_history(history):
        plt.plot(history.history['acc'])
        plt.plot(history.history['val_acc'])
        plt.title('model accuracy')
        plt.ylabel('accuracy')
        plt.xlabel('epoch')
        plt.legend(['train', 'val'], loc='upper left')
        plt.show()
```

ГРАФИК ОБУЧЕНИЯ

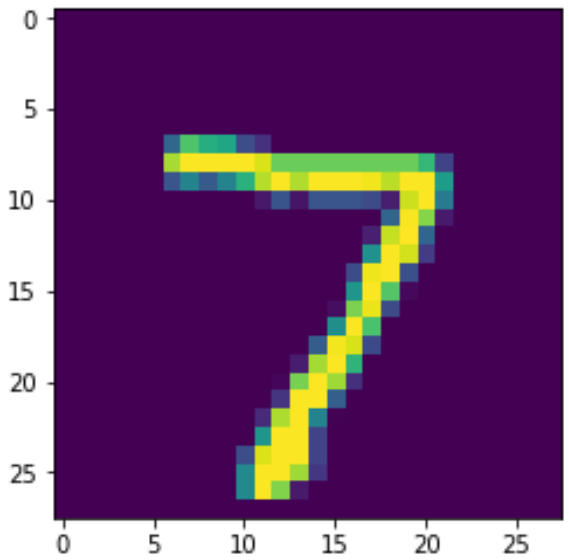
```
[35]  plot_history(history)
```



КАК ИСПОЛЬЗОВАТЬ

```
plt.imshow(x_test[0,:,:,0])
pred_class = model.predict_classes(x_test[0].reshape(-1, 28, 28, 1))
print(pred_class)
```

```
[7]
```



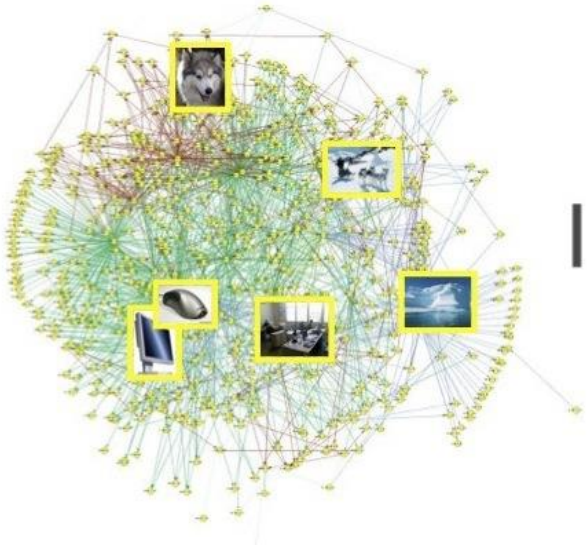
КАК СОХРАНИТЬ

```
model.save('my_model.h5')

saved_model = keras.models.load_model('my_model.h5')
pred_class = saved_model.predict_classes(x_test[0].reshape(-1, 28, 28, 1))
print(pred_class)
```

```
[7]
```

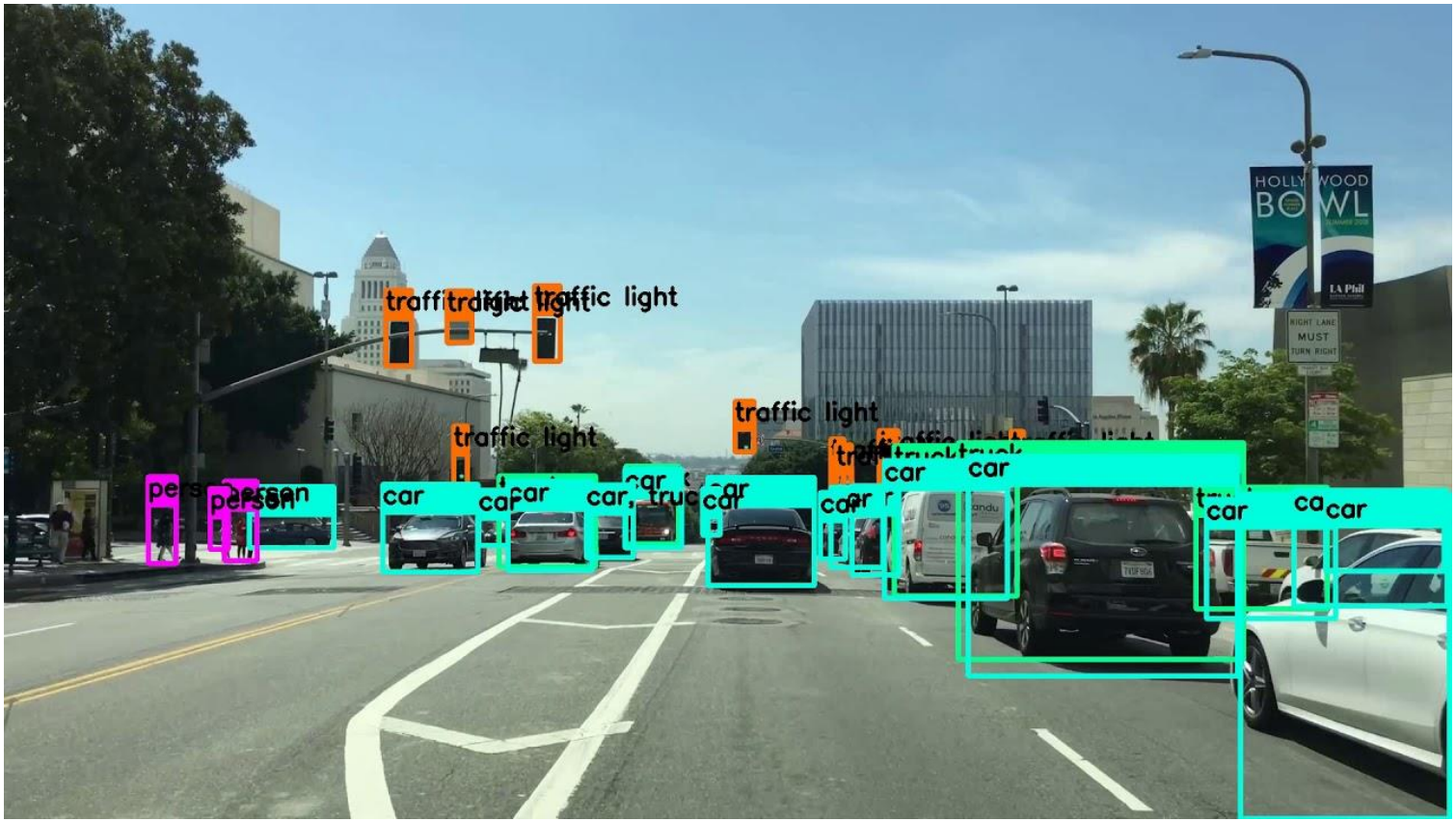
А ЧТО КРОМЕ ЦИФР?

Что угодно – современные датасеты содержат до 1000 самых разных классов, которые нейросети умеют различать.

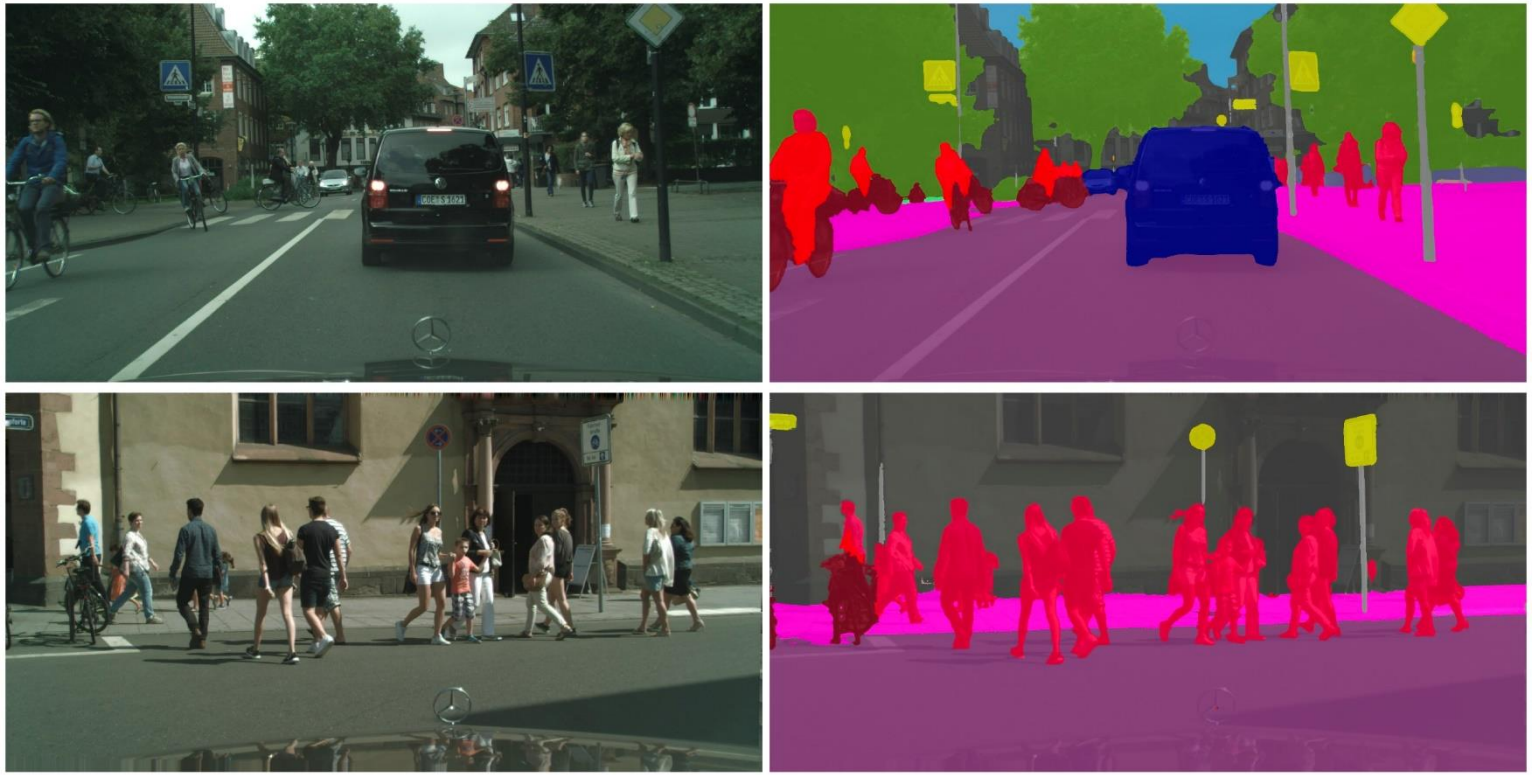


IMAGENET

ДЕТЕКТОРЫ

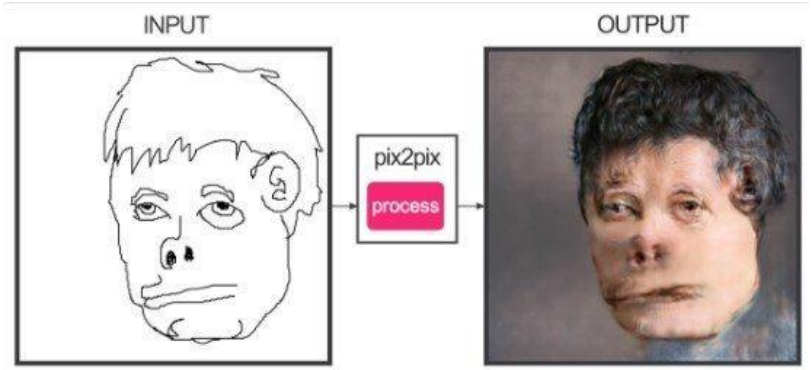

traffic light
traffic light
traffic light
car
car
car
car
car
car
car
car
car
car
car
car
car
HOLLYWOOD BOWL
LA Phil
RIGHT LANE MUST TURN RIGHT

СЕГМЕНТАЦИЯ

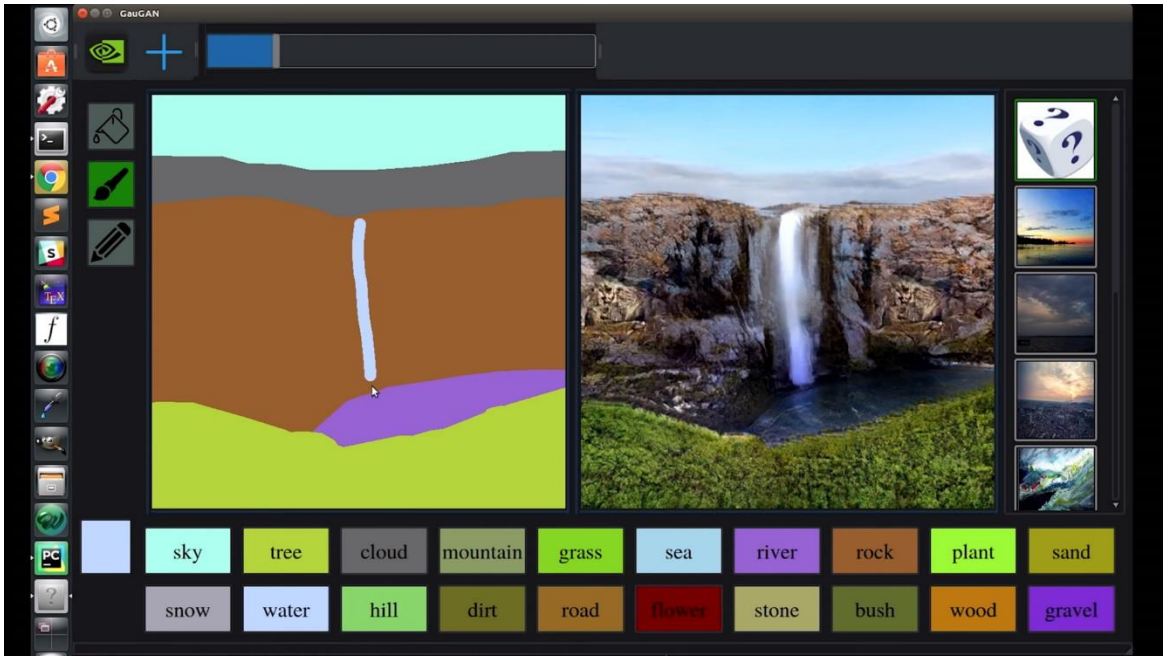


ГЕНЕРАЦИЯ

GAN – Generative adversarial network



GAN'S BY NVIDIA



DEEPFAKE

КРОМЕ ЭТОГО

Детектирование объектов на фото/видео

Генерация текстов

Машинный перевод

Написание и поиск музыки

Диагностика болезней

Предсказание погоды

Персональные рекомендации

…

ДОМАШНЕЕ ЗАДАНИЕ

1) Поиграться со ~~шрифтами~~ слоями, их параметрами

2) Обучить свою нейросеть на датасете fashion mnist

(**from** keras.datasets **import** fashion_mnist)

3) Если это окажется просто, то попробуйте cifar10, cifar100

Учтите, что в cifar – RGB изображения, поэтому в местах, где идет речь о размерности данных, нужно будет поменять с 1 на 3