1. **Недостатки полносвязной многослойной сети. Свёрточные нейронные сети. Принципы свёрточных сетей.**

**Сверточные нейронные сети**

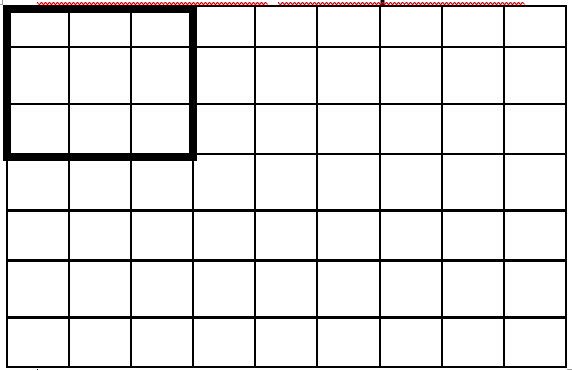
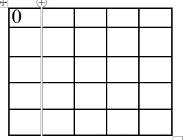
Недостатки полносвязной многослойной сети для анализа изображений:

* Большое количество весов (>600000 весов)
* Обучение требует больших объёмов данных
* Пространственная структура данных игнорируется

Принципы свёрточных сетей:

* Локальное восприятие
* Разделяемые веса
* Уменьшение размерности

Локальное восприятие



Сверточный слой

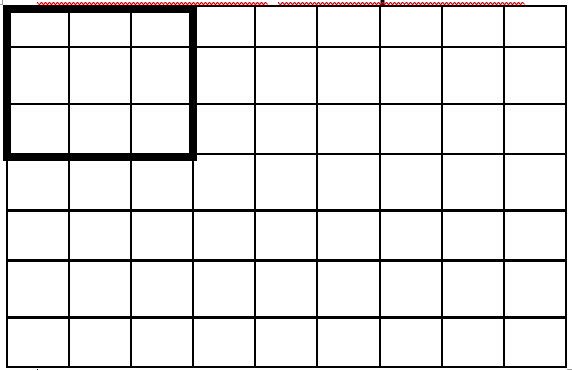
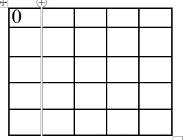
Входные сигналы

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1/9 | 1/9 | 1/9 |  | 0 | -1 | 0 |  | 0 | -1 | 0 |
| 1/9 | 1/9 | 1/9 |  | -1 | 4 | -1 |  | -1 | 5 | -1 |
| 1/9 | 1/9 | 1/9 |  | 0 | -1 | 0 |  | 0 | -1 | 0 |
| размытие | | |  | Выделение границ | | |  | Повышение четкости | | |

В нейронных сетях ядра свертки определяются

автоматически в процессе обучения.

**Разделяемые веса**



0

Сверточный слой

Входные сигналы

**Уменьшение размерности**

* Распознавание объектов вне зависимости от масштаба
* Факт наличия признака важнее Знания места его точного положения на изображении
* Слои подвыборки
* Усреднение
* Выбор максимального значения

1. **Архитектура свёрточных сетей. Преимущества. Свёрточная сесть Lenet-5**

**Свёрточные нейронные сети**

Архитектура свёрточных сетей

* Свёрточные слои
* Слои подвыборки

Преимущества

* Учитываются локальные особенности изображении
* Небольшое количество параметров для обучения

**LeNet 5** состоит из пяти обучаемых слоев - трех сверточных (карты признаков ( feature map) размеров 6, 16 и 120)

и двух полносвязных. Отсюда и его название LeNet 5.

Каждый сверточный слой имеет ядра размером 5 на 5 и шагом 1. Сверточные слои разделены активацией гиперболического тангенса, слоем среднего пулинга с размером ядра 2 на 2. Вывод последней свертки имеет размер (1, 1, 120), который ведет дальнейшему полносвязному соединению.

**[5х5 свертка,6] – [подвыборка2] –[5х5 свертка,16] –[ подвыборка2]-[ 5х5 свертка,120]-[полносвязный слой,84]-[полносвязный слой,10]**

1. **Распознавание объектов на изображениях, постановка задачи, данные для обучения. Пример кода создания архитектуры свёрточной сети для распознавания объектов на изображениях.**

Задача распознавания объектов

Распознавание объектов на изображениях

* Набор данных CIFAR-10

Изображения в **CIFAR-10**

* Размер 32х32
* Цветные изображения (коды интенсивности RGB)
* Набор данных для обучения - 50 000 (5 000 для каждого класса)
* Набор данных для тестирования - 10 000
* На каждом изображении только один объект

#задаeм seed для повторяемости результатов

numpy. random. seed (42)

#Загружаем данные

(X\_train, Y\_train), (X\_test, Y\_test)=cifar10.load\_dat

a()

#Нормализуем данныe

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype(' float32')

X\_train /= 255

X\_test /= 255

#прeобразуeт мeтки в катeгории

Y\_train = np\_utils.to\_categorical(y\_train, 10)

Y\_test =np\_utils.to\_categorical(y\_test, 10)

# создаем последовательную модель

model =Sequential ()

первый сверточный слой

model.add(Conv2D (32, (3, 3), padding=' same'

input\_shape=(32, 32, 3), activation=’relu'))

# Второй сверточный слой

model.add(Conv2D(32, (3, 3), activation='relu', padding='same’))

# первый слой подвыборки

model.add(MaxPooling2D (pool\_size= (2, 2)))

# Слой pегуляризации Dropout

model.add(Dropout (0.25))

1. **Техника предотвращения переобучения свёрточных сетей. Пример использования.**

**Dropout**

* Проблема переобучения в сверточной сети
* Нейроны настраиваются на совместное выделение нужных признаков
* Учитываются особенности конкретной выборки
* **Dropout** - техника предотвращения переобучения
* При предъявлении каждого объекта обучения, нейроны с
* заданной вероятностью отключаются
* Оставшиеся нейроны обучаются распознавать необходимые
* признаки без участия соседних

# третий сверточный слой

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

# Четвертый сверточный слой

model.add(Conv2D(64, (3, 3), activation='relu'))

# второй слой подвыборки

model.add(MaxPooling2D (pool\_size=(2, 2)))

: Слой регуляризации Dropout

model.add (Dropout (0.25))

* Классификатор

#Слой преобразования данных из 2D представления в плоское

model.add(Flatten())

#полносвязный слой для классификации

model.and(Dense(512, activation='relu'))

#слой peгyляризации Dropout

model.add(Dropout(0 5))

#Выходной полносвязный слой

model.add (Dense(10, activation='softmax')

1. **Пример кода обучения свёрточной сети для распознавания объектов на изображении.**

# задаем параметры оптимизации

model. compile(loss=' categorical cross entropy',

optimizer='sgd',

metrics=[' accuracy'])

#обучаем модель

model.fit(x train, train,

batch size=32,

epochs=25,

validation\_split=0.1,

shuffle=True,

verbose=2)

#оцениваем качество обучения модели на тестовых данных

scores = model.evaluate(X\_test, Y\_test, verbose=0)

print("точность работы на тестовых данных: %.2f%%"

%(scores [1] \*100))

1. **Перенос обучения. Преимущества, примеры предварительно обученных нейронных сетей и их характеристика.**

**Перенос обучения**

**Преимущества использования предварительно обученных нейронных сетей:**

* Архитектуры нейронных сетей, решающих реальные задачи, часто состоят из десятков слоев и имеют миллиарды параметров, которые необходимо обучить
* Обучение нейронной сети может занимать дни и месяцы. И это обучение с одним заранее заданными гиперпараметрами
* К тому же не у всех ест достаточные ресурсы

Перенос обучения – использование предварительно обученной сети для других задач

Предварительно обученные нейронные сети в **Keras**

VGG16, VGG19, inception v3, ResNet50, Xception и др.

Набор данных для обучения ImageNet

1000 типов объектов

**Обзор существующих предварительно обученных нейронных сетей**

* **VGG**
* Глубокая нейронная свёрточная сеть, имеющая последовательное применение слоев и из-за этого все же ограниченная по глубине.
* Имеет две версии: 16 и 19 слоёв
* **ResNet**
* Использует непоследовательные преобразования, периодически объединяя (суммированием) выход текущего слоя с выходом слоя, который был обработан на несколько шагов ранее.
* **Inception**
* Основана на блоках с древовидной архитектурой.
* Этот блок параллельно применяет к слою несколько операций, а затем агрегирует всю информацию снова в один слой.
* **Dance Net**
* Состоит из «плотных» блоков.
* Каждый блок принимает на вход все предыдущие карты признаков.
* При этом объединение карт, в отличие от Reset, где происходит суммирование карт признаков, происходит конкатенированием
* **Exception**
* Отличается тем, что отдельно выполняет обработку и преобразование канальной информации, а отдельно -пространственной.
* Канальная информация извлекается свертками 1x1, а пространственная выделяется сверткой 3х3, но к каждому каналу по отдельности

1. **Архитектура предварительно обученной нейронной сети VGG16. Основные этапы переноса обучения. Пример кода создания нейронной сети при переносе обучения.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3х3 conv, 64 | 3х3 conv, 64 | **pool/2** | 3х3 conv, 128 | 3х3 conv, 128 | **pool/2** | 3х3 conv, 256 | 3х3 conv, 256 | 3х3 conv, 256 | **pool/2** | 3х3 conv, 512 | 3х3 conv, 512 | 3х3 conv, 512 | **pool/2** | 3х3 conv, 512 | 3х3 conv, 512 | 3х3 conv, 512 | **pool/2** | fc 4096 | fc 4096 | fc 4096 |

**Conv**- сверточные слои; **Pool** – слои подвыборки; **Fc**- полносвязные слои

**Основные этапы переноса обучения:**

* Загрузка свёрточной части предварительно обученной нейронной сети
* Добавление к свёрточной части нового классификатора для своего набора изображений
* Заморозка веса в предварительно обученной части сети
* Обучение полученной составной сети на новом наборе данных
* Оценка качества обучения на тестовом наборе данных
* Разморозка нескольких свёрточных слоёв
* Выполнение тонкой настройки нейронной сети на новом наборе данных
* Оценка качества обучения на тестовом наборе данных

model = VGG16(weights='imagenet')

model summary()

Перенос обучения

files.upload()

filename ‘space plane 3pg'

img = image.load\_img(filename, target\_size (224, 224))

plt. imshow (img)

plt. show()

x = image img\_to\_array(img)

x = np. Expand\_dims(x, axis=0)

x = preprocess\_input(x)

press = model.predict(x)

preds

* Набор данных для обучения

train\_dataset = image\_dataset\_from\_directory('seg\_train/seg\_train'

subset='training',

seed=42,

validation\_split=0.1,

batch\_size=120,

image\_size=(120, 120))

validation dataset image\_dataset\_from directory('seg train/seg\_train'

subset= ‘validation'

seedm=42,

validation\_split=0.1,

batch\_size=120

image\_size (120, 120))

* Набор данных для тестирования

test\_datanet = image\_dataset\_from\_directory('seg\_test/seg\_test',

shuffle=True,

label\_mods='int',

batch\_size =128,

image\_size=(128, 128))

1. **Расширение данных для обучения. Слои Keras для дополнения данных. Пример кода для расширения данных.**

**Расширение данных для обучения**

Слой Keras для дополнения данных:

* **Randomflip** - отражение изображения по горизонтали или вертикали
* **RandomRotation** - поворот изображения
* **RandomCrop** - вырезание фрагмента изображения из случайного места
* **CenterCrop** - вырезание фрагмента из центра изображения

# Установить параметры генератора

datagen = image.ImageDataGenerator(fill\_mode='wrap', zoom\_range=[4, 4])

gen\_data = datagen.flow\_from\_directory(PATH,

batch\_size=1,

shuffle=False,

save\_to\_dir=SAVE\_PATH,

save\_prefix='gen',

target\_size=(224, 224))

# Создать 9 изображений

for i in range(9):

gen\_data.next()

# Найти локально сгенерированное изображение и напечатать 9 изображений на одной фигуре

name\_list = glob.glob(gen\_path+'16/\*')

fig = plt.figure()

for i in range(9):

img = Image.open(name\_list[i])

sub\_img = fig.add\_subplot(331 + i)

sub\_img.imshow(img)

plt.show()