DEEP LEARNING

BY M.J.PASSLAR

COMPUTERONIC - TEHRAN - IRAN

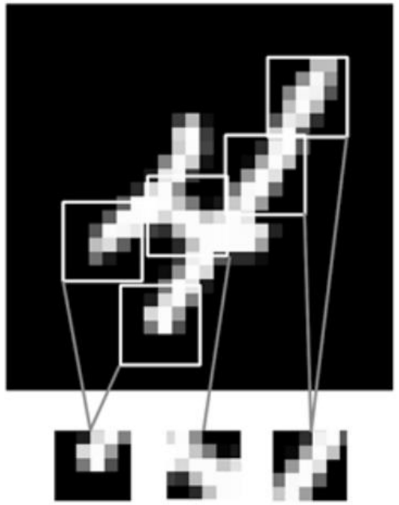
2023

CHAPTER 6 : CONVNETS(CNN)

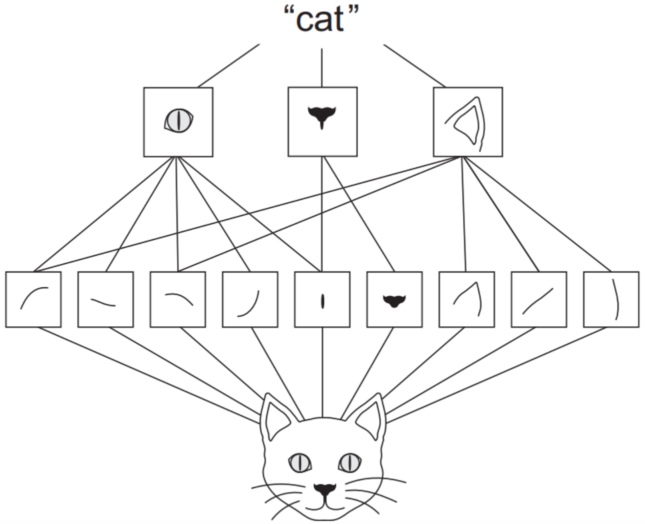
Convolutional networks (convnets)

- Convolutional networks use for machine vision problems
- What's different between Dense and Conv layers ?

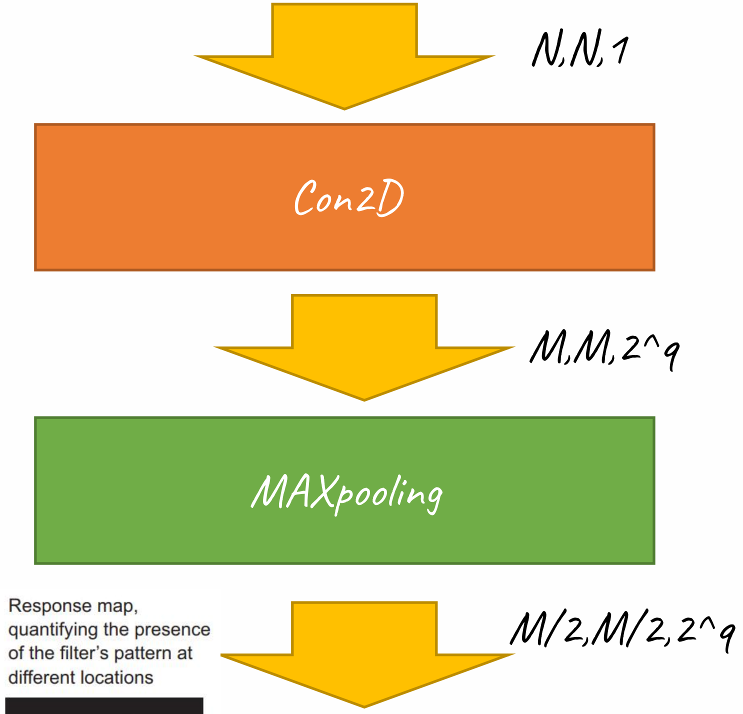
  Dense layers try to find global pattern in the data but Conv layers try to find out local pattern.



- Conve layers give you two critical options :
  - The pattern they learn are translation invariant
  - They can learn spatial hierarchies of patterns



Standard structure of Convnets



N,N,1

Con2D

M,M,2^q

MAXpooling

M/2,M/2,2^q

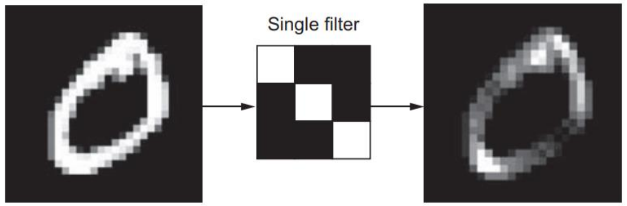
Extract feature extraction

Compressed Extracted features

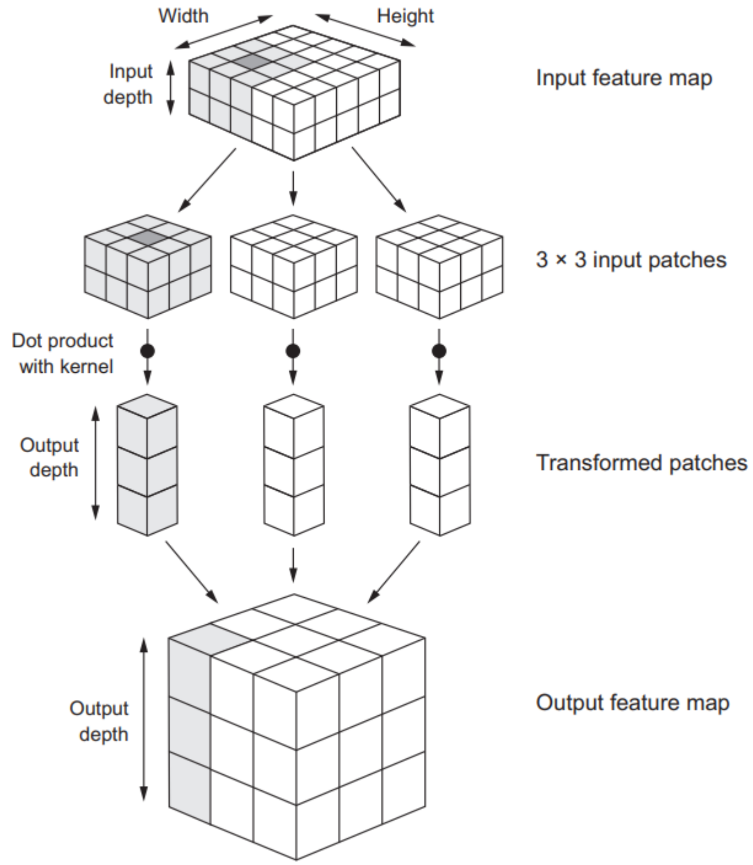
Gray scale image with the size of N*N

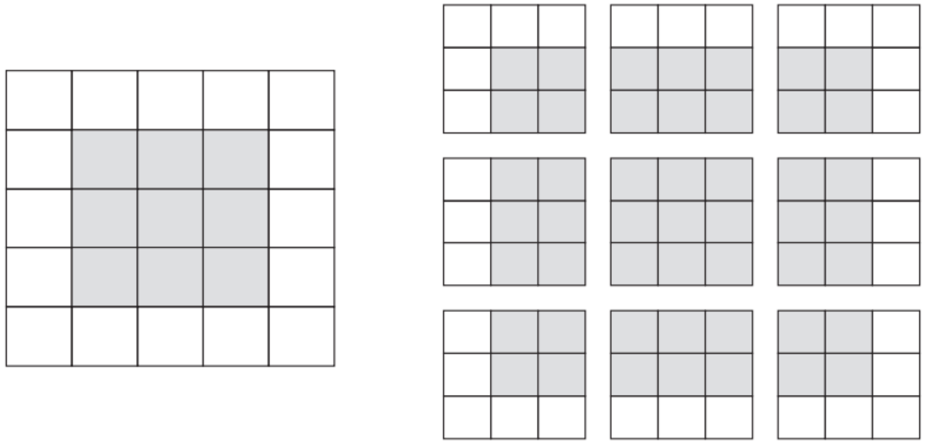
2^q filter (to extract features)
M*M -> filter response

Sample down results
Reduce response by 2

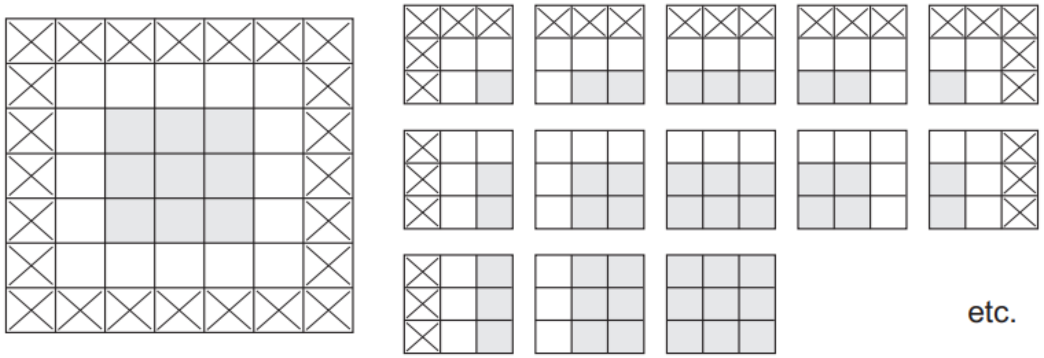

Original input
Single filter
Response map, quantifying the presence of the filter's pattern at different locations

Conv2D





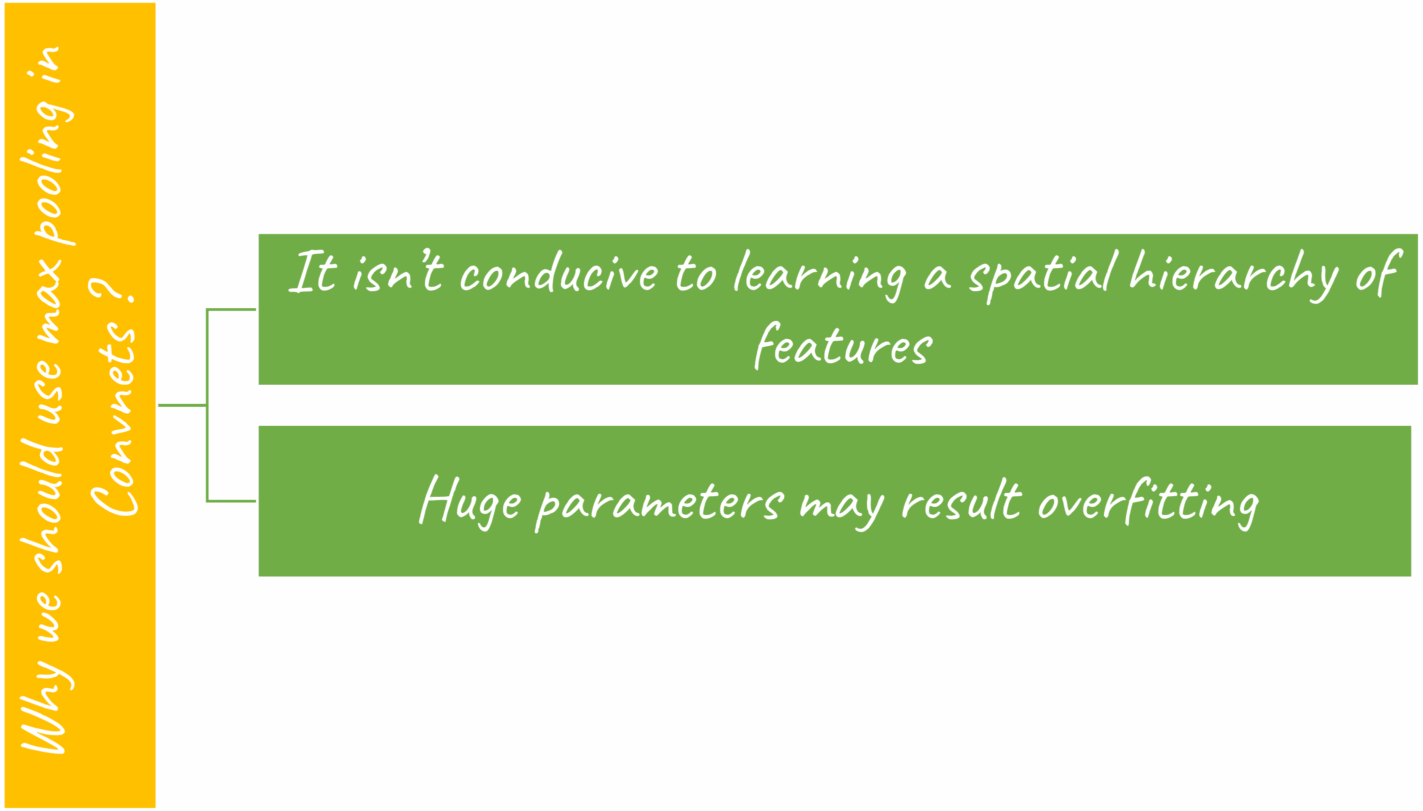
Border



Padding

MAXpooling

Conv2D layers have a huge parameters to calculate so we need to reduce them with Maxpooling layers



Why we should use max pooling in Convnets ?

It isn't conducive to learning a spatial hierarchy of features

Huge parameters may result overfitting

MNIST with Convnets

```
#preparing MNIST
from keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

Create and compile your model

```
#creat and compile your model
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

Model structure

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 flatten (Flatten)           (None, 576)               0

 dense (Dense)               (None, 64)                36928

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

explain it ...

Evaluate your model

```
#compile and evaulate your model

model.fit(train_images, train_labels, epochs=5, batch_size=64)
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

Do you remember the Solution for MNIST with Dense layers ?

97%

Dense layer network

99%

Conv2D and Maxpooling network

Grade up problems …

- Use LB-processing workspace
- Download Dog and Cat data set from Kaggle :
  - https://www.kaggle.com/c/dogs-vs-cats/data
  - 854 Mbyte
  - 25,000 dogs and cat images (dog = 1 , cat = 0)
  - In all situation and variety places
  - Serious problem from 2013 up to know