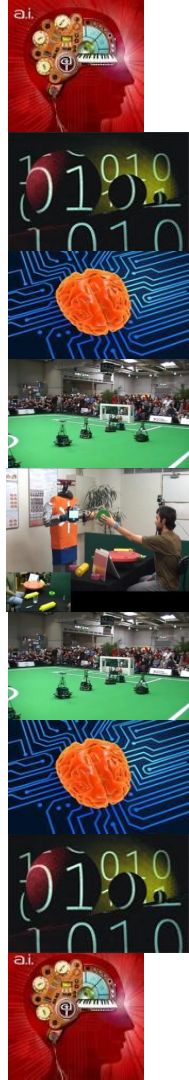
Convolution Neural Networks

COS 314

Outline

- History of Convolution Neural Network
- Application Area
- Structure
- Functionality
- Future



Introduction

- Convolutional Neural Networks or CovNets
- Alex Krizhevsky won the ImageNet Challenge – AlexNet
- Image classification
- Classification

  Class

  Probabilities for each class

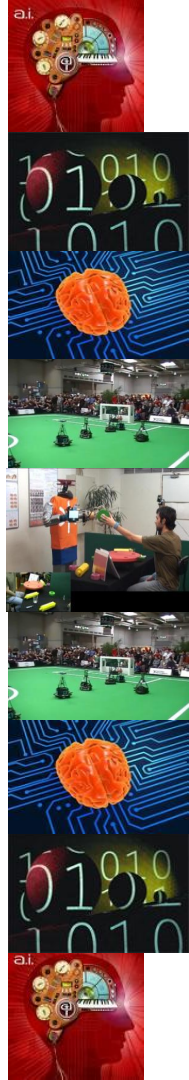


Image Classification

- Image converted to a matrix of pixels
- Grayscale images – a single integer 2D matrix with values in the range 0 to 255
- Colour images – 3 stacked (RGB) 2D matrices with values in the range 0 to 255
- Size of the matrix is resolution dependent
- Preprocessing to reduce the size of the matrix

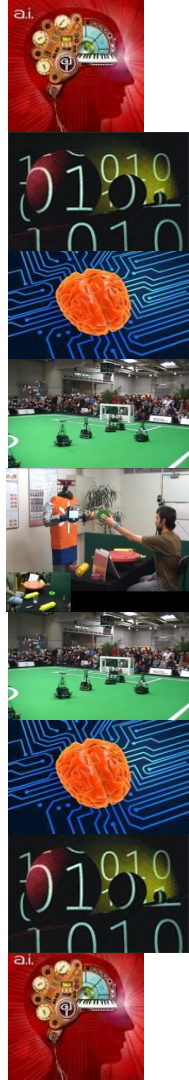
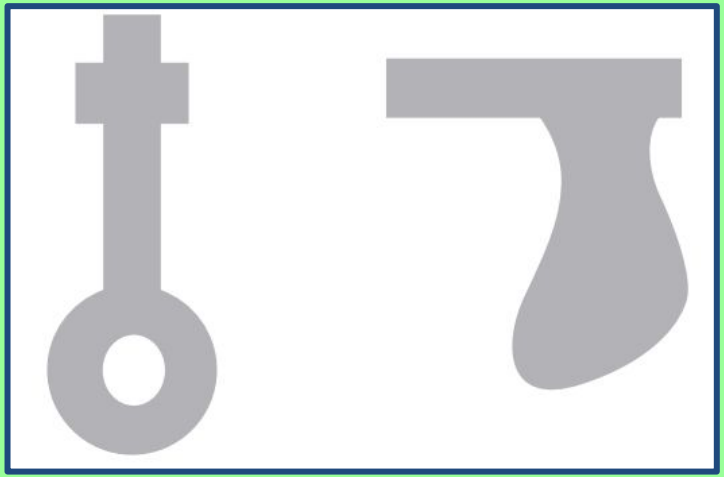


Image Representation



```
1 1 1 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 2 2 2 2 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1
1 1 1 1 2 2 2 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1
1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 1 1 1 1
1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 1 1 1
1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 1 1
1 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 1 1
1 1 2 2 4 4 2 2 2 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 1 1 1 1
1 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1
1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Image Representation

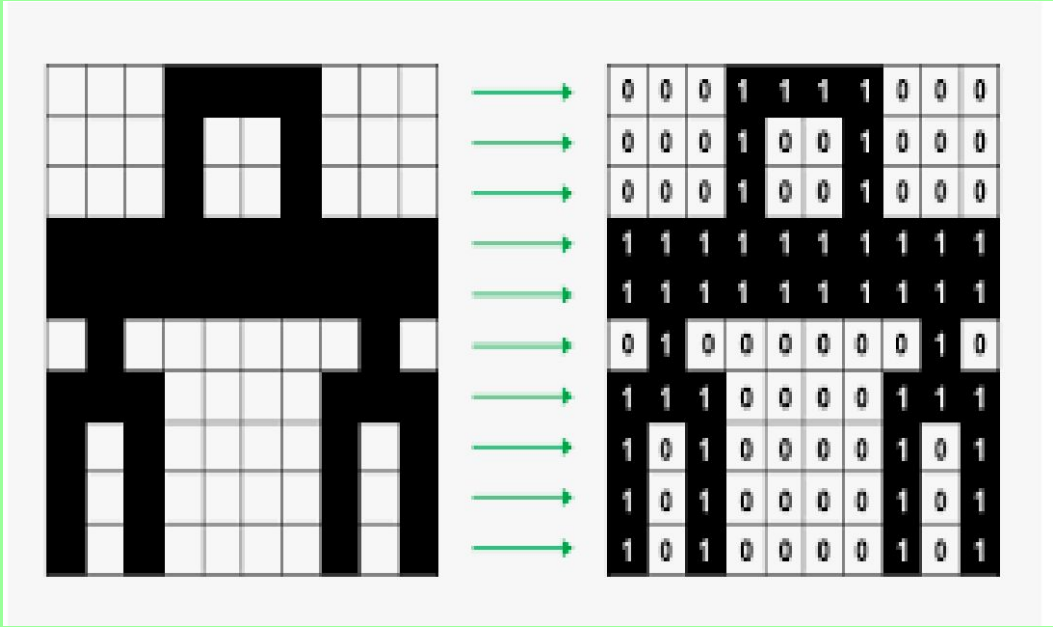
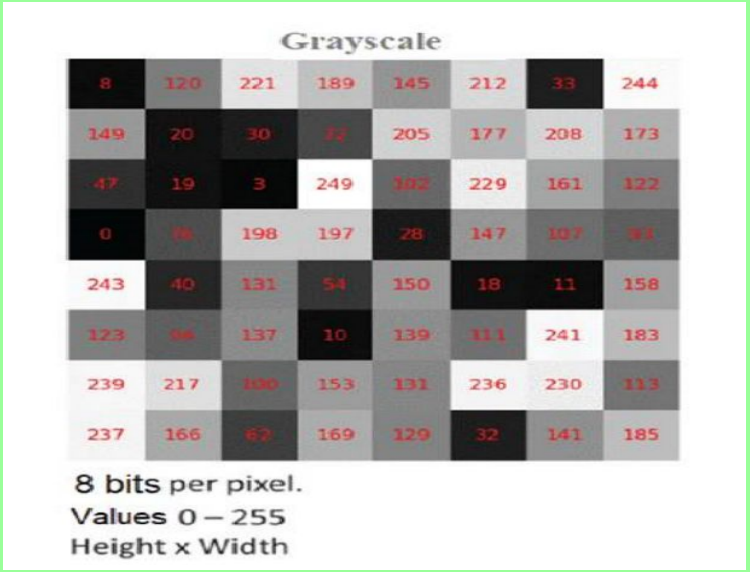


Image Representation



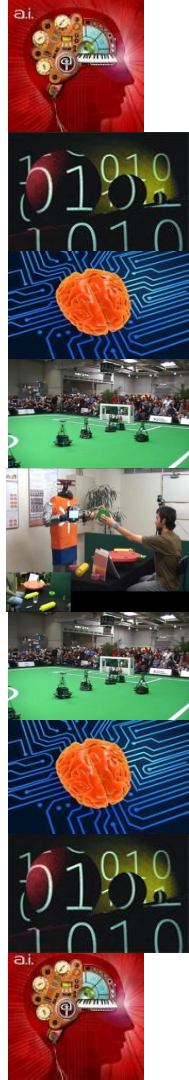


Image Representation

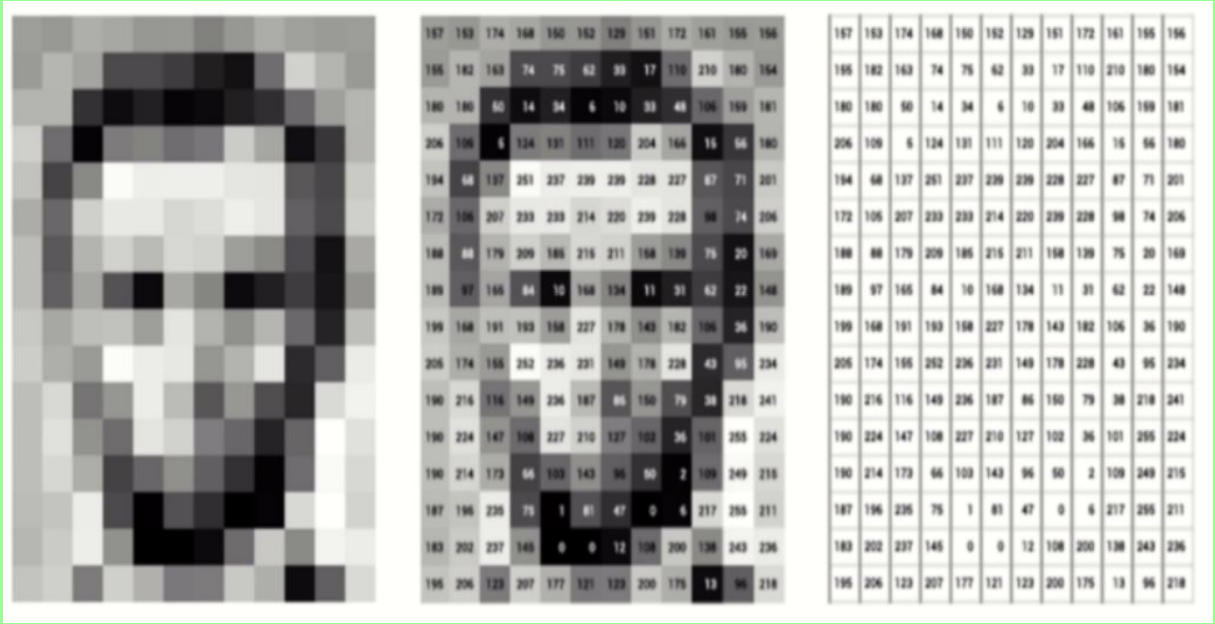
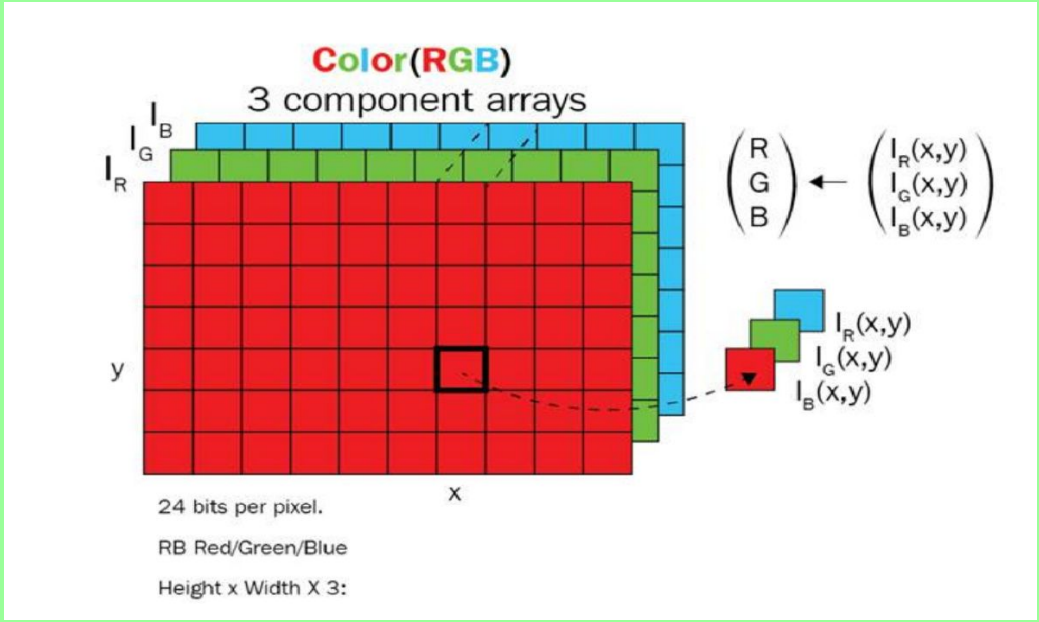
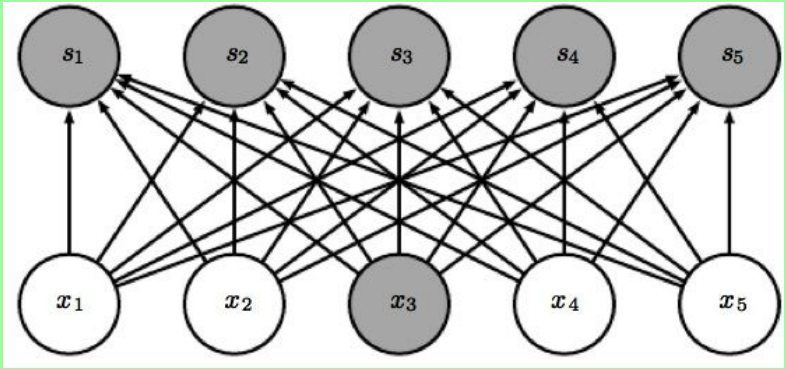
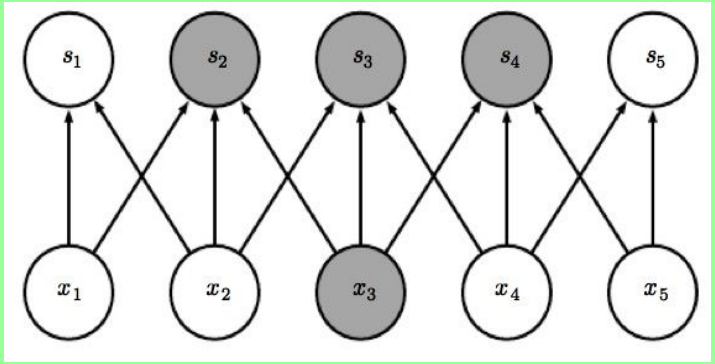


Image Representation



Fully Connected vs Sparse Connected

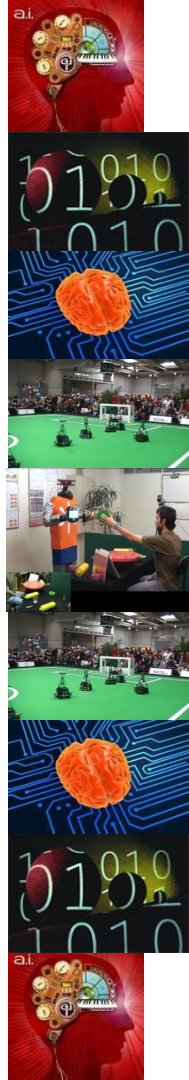




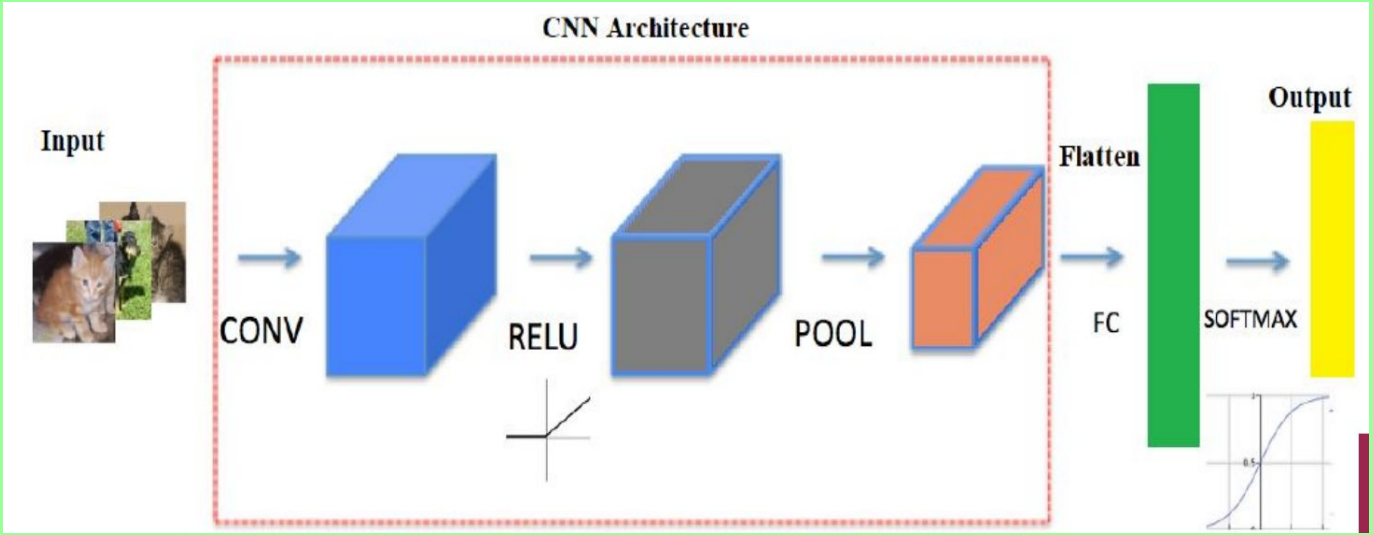
Parameter Sharing

CNN Layers

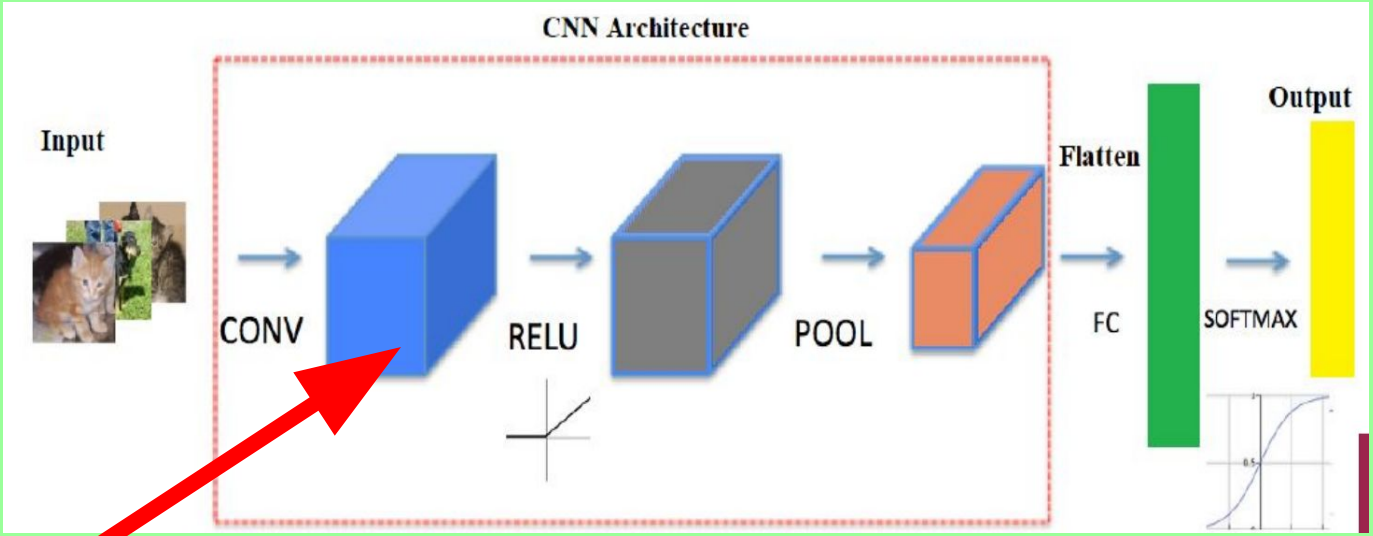
- Convolutional layer
- Nonlinear or ReLU layer
- Pooling layer
- Fully connected layer



Overview

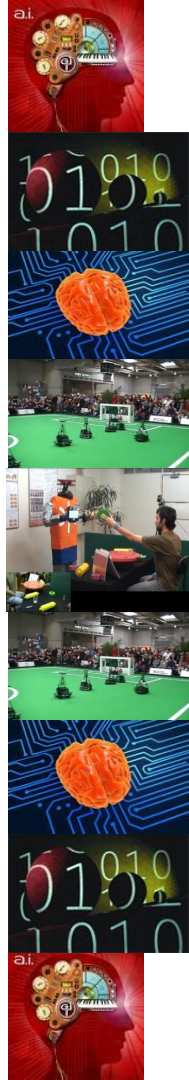

CNN Architecture
Input
CONV
RELU
POOL
Flatten
FC
SOFTMAX
Output

Convolution Layer



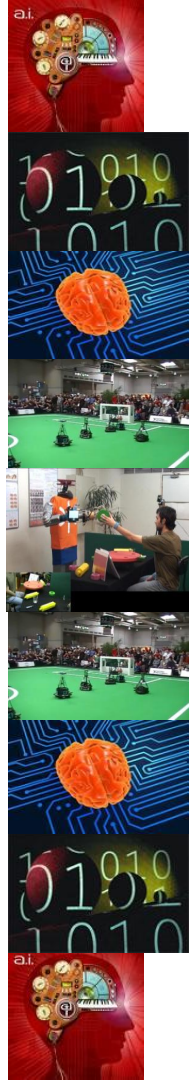
Convolution Layer

- First layer after the input layer
- Additional convolutional layers
- Convolution operator
- Filter or kernel
- Feature map
- Sparsely connected layer – parameter sharing



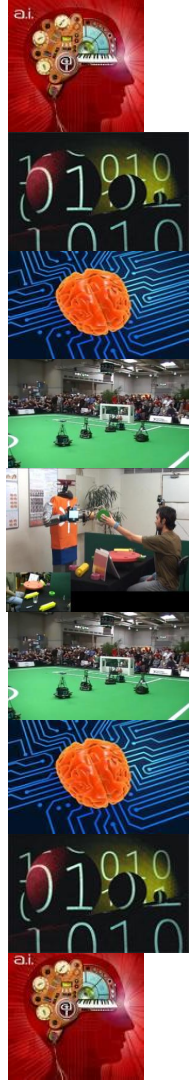
Convolution Layer

- Different convolution operators include:
  - Identity
  - Edge detection
- Different convolutions are used with different kernels/filters.
- Selection of values for kernels/filters
- Stride
- Zero padding



Determining the Feature Map

- Decide on the dimensions of the kernel, e.g. 3x3
- Determine the values of the kernel.
- Apply the convolution to produce the feature map.
- Map the kernel across the pixel matrix to produce the feature map.



Determining the Feature Map -2D Example

Input Matrix

0	1	0	1	0	0
0	1	1	0	0	0
1	0	1	1	0	1
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	0	1

6X6

*

Filter/Kernel

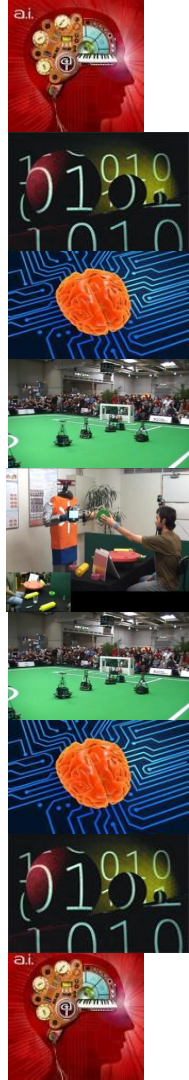
0	0	0
1	0	1
-1	0	-1

3X3

=

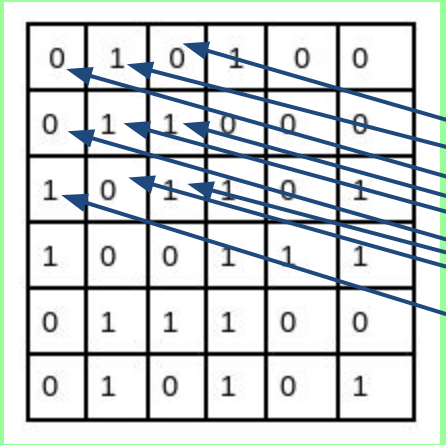
Output Feature Map

4X4



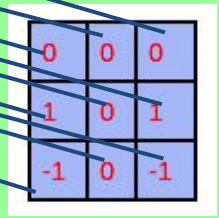
Determining the Feature Map

Input Matrix



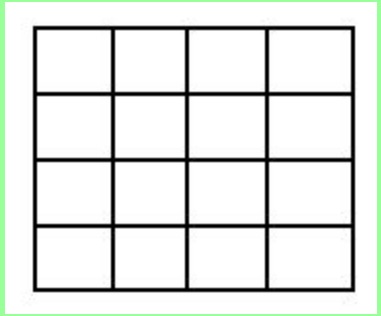
*

Filter/Kernel



=

Output Feature Map



Determining the Feature Map

Input Matrix

0	1	0	1	0	0
0	1	1	0	0	0
1	0	1	1	0	1
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	0	1

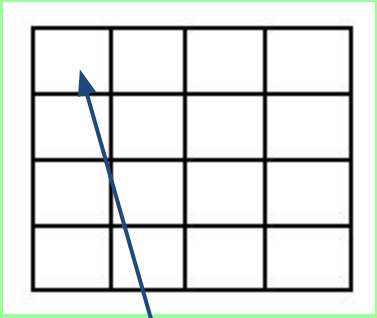
*

Filter/Kernel

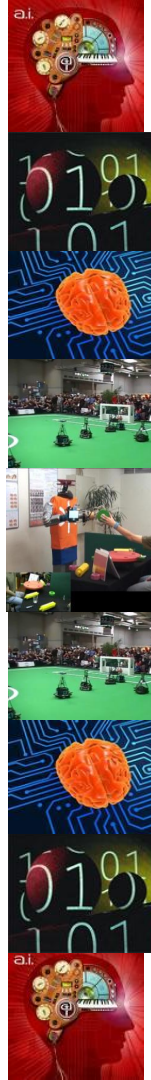
0	0	0
1	0	1
-1	0	-1

=

Output Feature Map



Convolution = 0*0 + 1*0 + 0*0 + 0*1 + 1*0 + 1*1 + 1*-1+ 0*0+ 1*-1 = **-1**



Determining the Feature Map

Input Matrix

0	1	0	1	0	0
0	1	1	0	0	0
1	0	1	1	0	1
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	0	1

*

Filter/Kernel

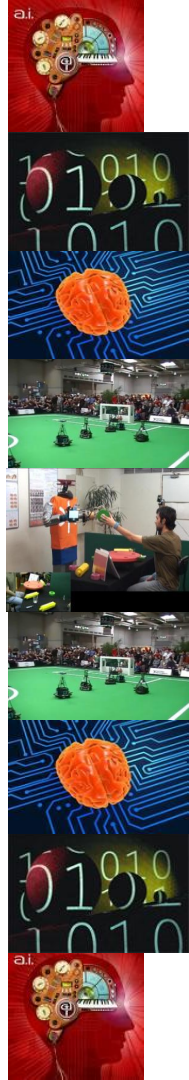
0	0	0
1	0	1
-1	0	-1

=

Output Feature Map

-1			

Convolution = 1*0 + 0*0 + 1*0 + 1*1 + 1*0 + 0*1 + 0*-1+ 1*0+ 1*-1 = **0**



Determining the Feature Map

Input Matrix

0	1	0	1	0	0
0	1	1	0	0	0
1	0	1	1	0	1
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	0	1

*

Filter/Kernel

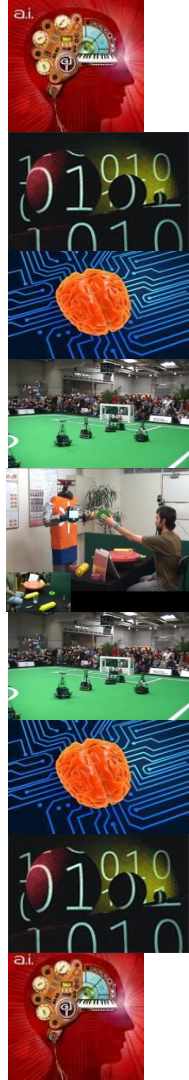
0	0	0
1	0	1
-1	0	-1

=

Output Feature Map

-1	**0**		

Convolution = 0*0 + 1*0 + 0*0 + 1*1 + 0*0 + 0*1 + 1*-1+ 1*0+ 0*-1 = **0**



Determining the Feature Map

Input Matrix

0	1	0	1	0	0
0	1	1	0	0	0
1	0	1	1	0	1
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	0	1

*

Filter/Kernel

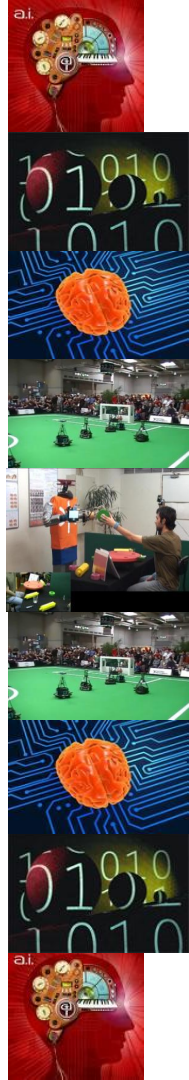
0	0	0
1	0	1
-1	0	-1

=

Output Feature Map

-1	**0**	**0**	

Convolution = 1*0 + 0*0 + 0*0 + 0*1 + 0*0 + 0*1 + 1*-1+ 0*0+ 1*-1 = **-2**



Determining the Feature Map

Input Matrix

0	1	0	1	0	0
0	1	1	0	0	0
1	0	1	1	0	1
1	0	0	1	1	1
0	1	1	1	0	0
0	1	0	1	0	1

*

Filter/Kernel

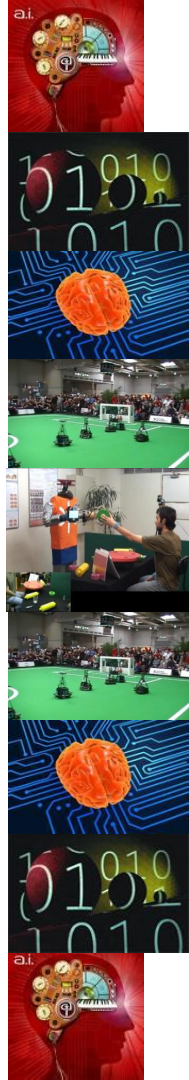
0	0	0
1	0	1
-1	0	-1

=

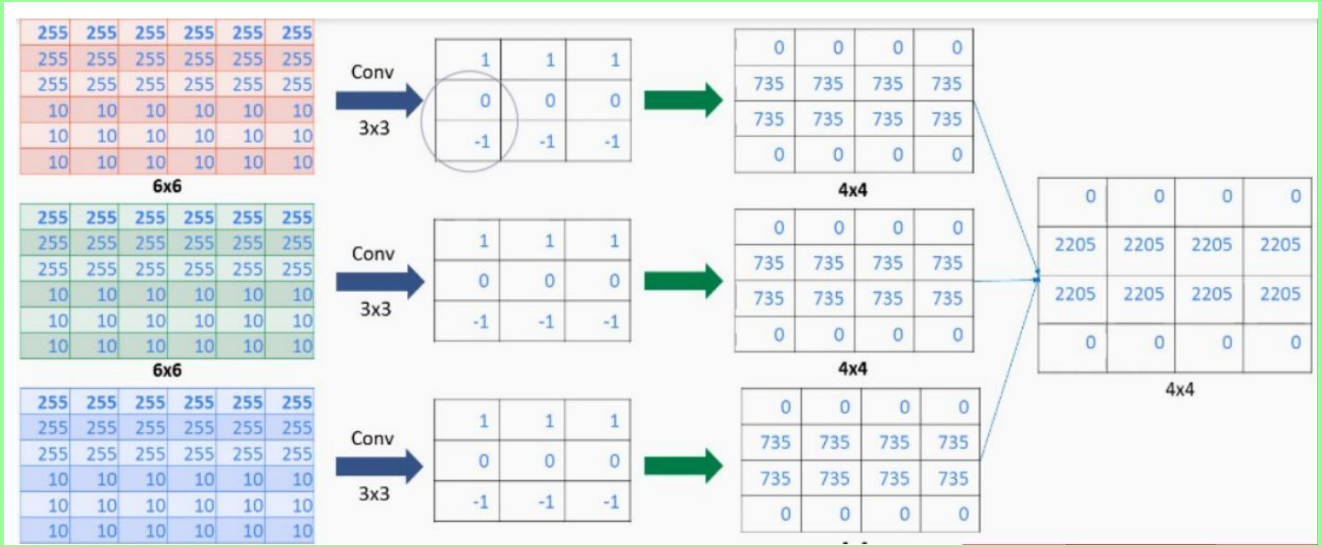
Output Feature Map

-1	0	0	-2
-1			

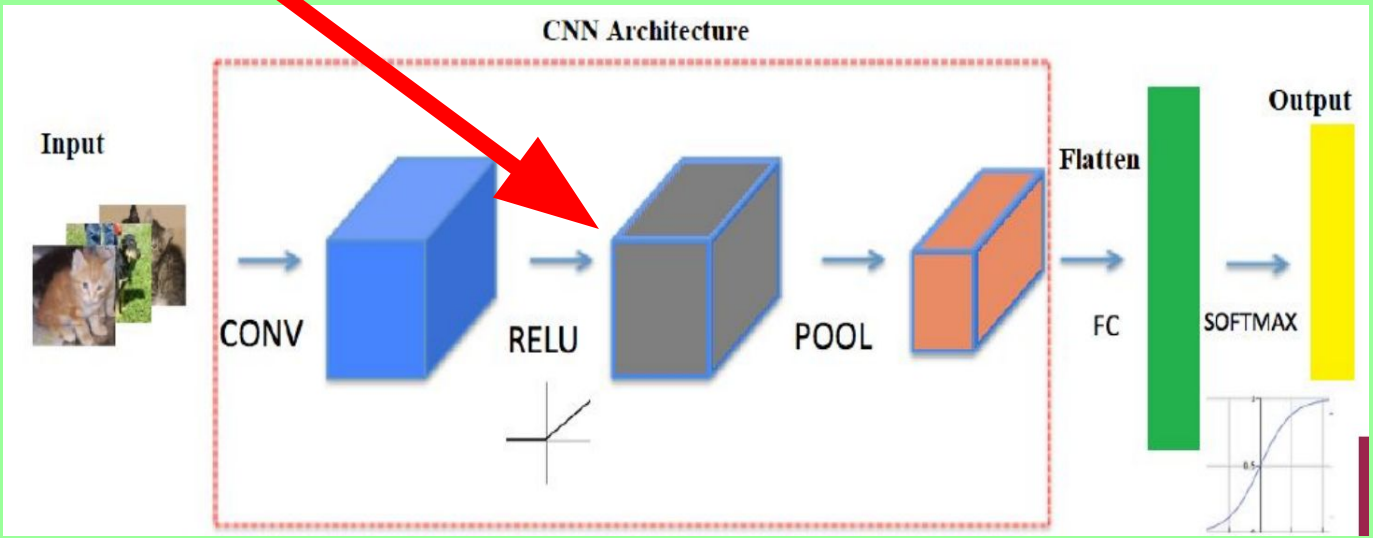
Convolution = 0*0 + 1*0 + 1*0 + 1*1 + 0*0 + 1*1 + 1*-1+ 0*0+ 0*-1 = **-1**



Convolution Layer - 3D Example

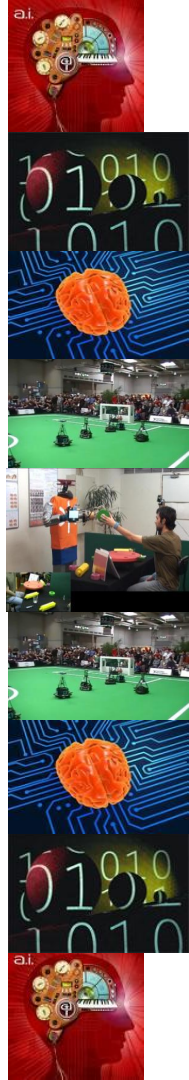


ReLU Layer

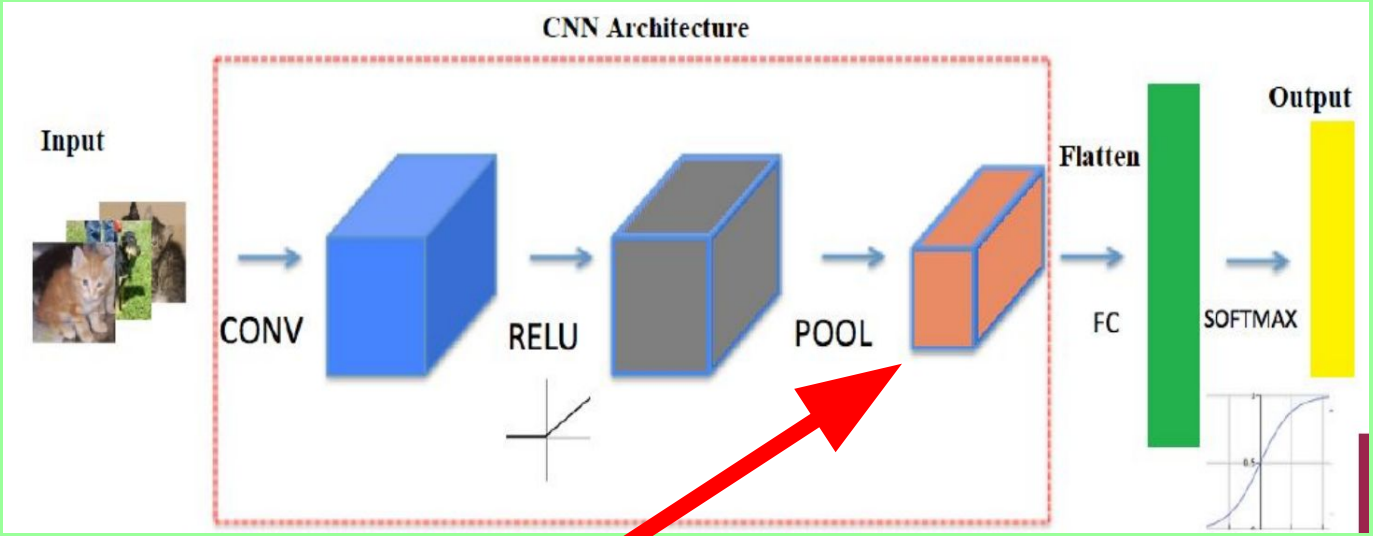

CNN Architecture
Input
CONV
RELU
POOL
Flatten
FC
SOFTMAX
Output

Nonlinear ReLU Layer

- Is used to incorporate nonlinearity
- The ReLU activation function is used
- $f(x) = \max(0, x)$

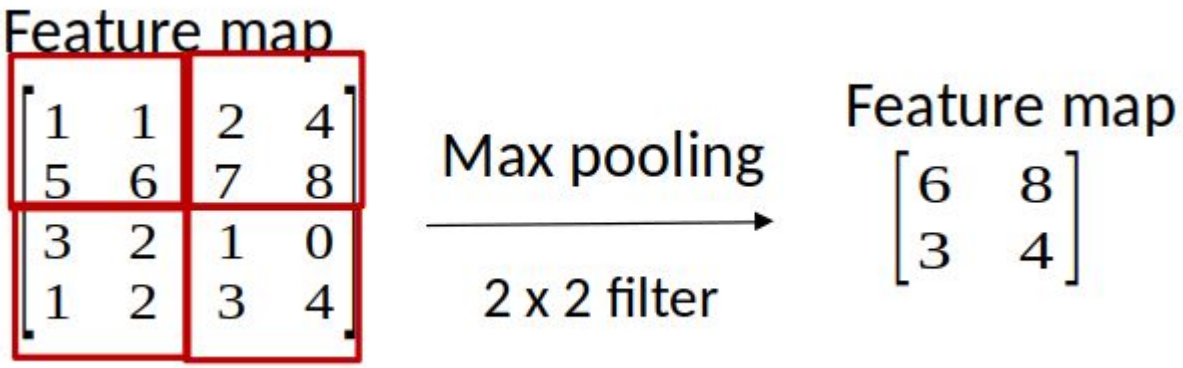


Pooling Layer

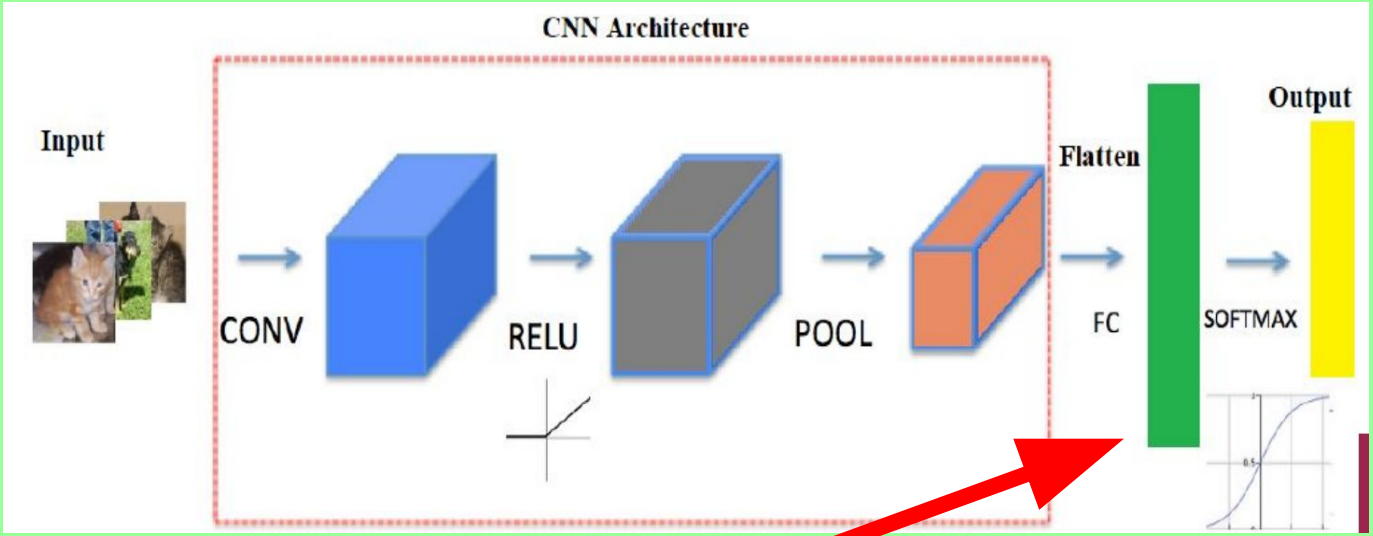


Pooling Layer

- Reduces the number of parameters
- This is achieved by reducing the size of the feature map

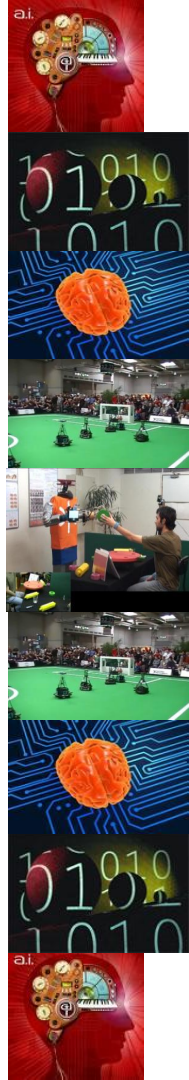


Fully Connected Layer

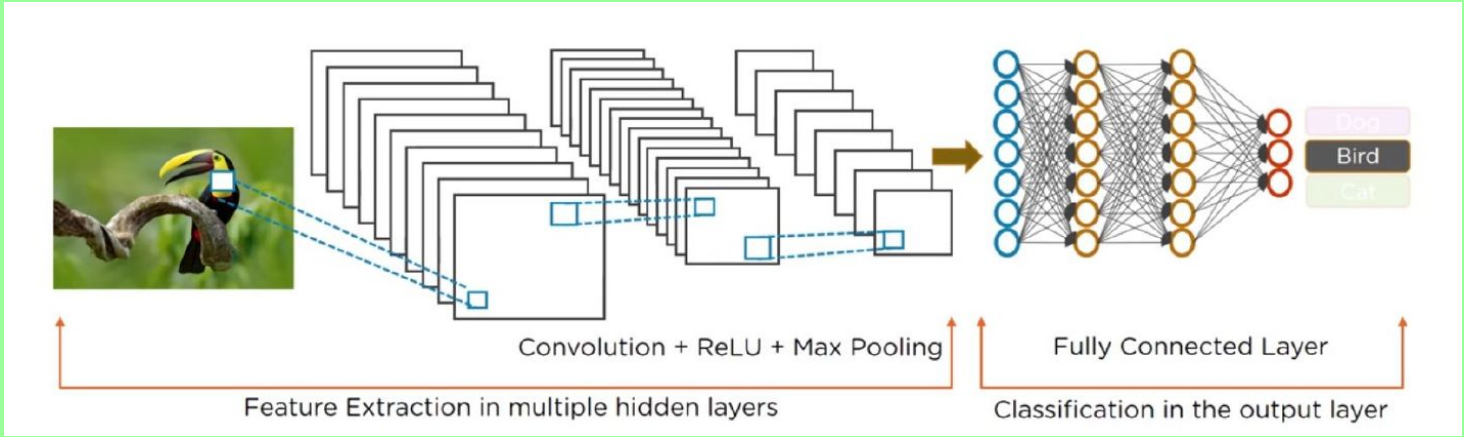


Fully Connected Layer

- Output layer
- Last or last two layers
- Softmax activation function
- Transfer learning



Fully Connected Layer



CNN - Architectures

- LeNet
- AlexNet
- GoogLeNet
- VGGNet
- RestNet
- DenseNet
- ImageNet and transfer learning

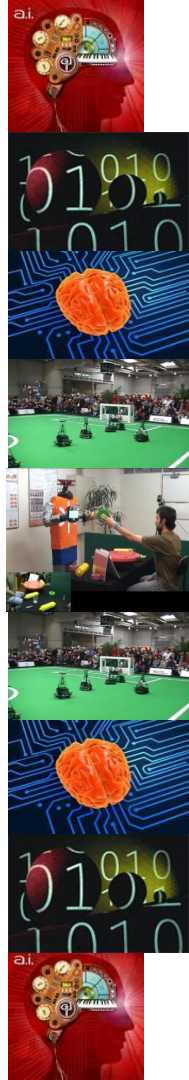
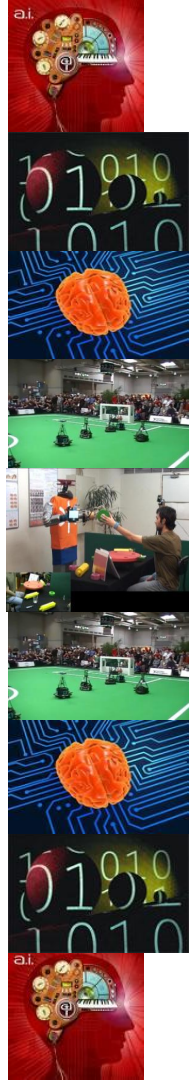


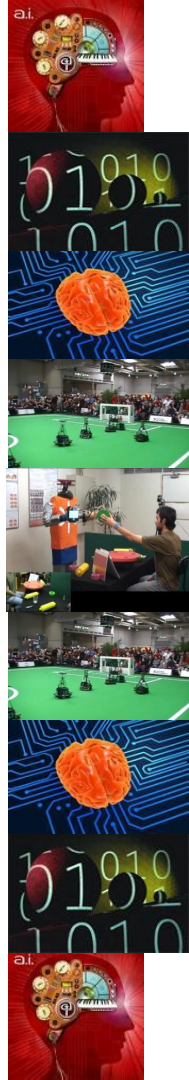
Image Classification -Transformers

- Use **Attention**
- Natural Language Processing based on RNN
- Vision Transformer performs better than CNNs

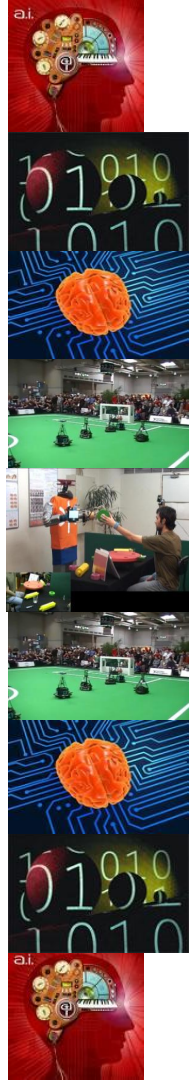


Multi-Task Learning

- Multi-task learning (MTL) is a machine learning technique where a model is trained on multiple learning tasks simultaneously.
- The goal is to improve the performance of each task.
- Through leveraging the knowledge and information shared across them.

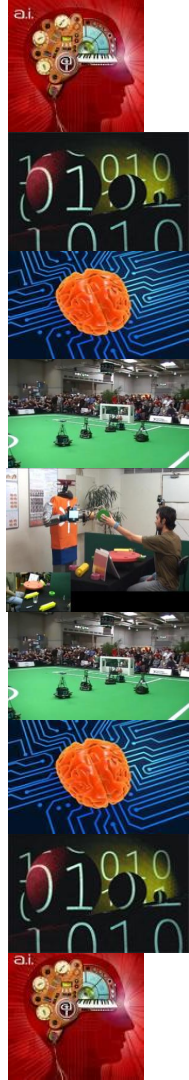


- Training a single model on multiple related tasks, the model can learn shared representations and features that are beneficial for all tasks.
- This can lead to better performance compared to training separate models for each task.



Advantages of Multitask

- **Improved Generalization:** By considering information from multiple tasks, the model can potentially generalize better to unseen data within each task domain.
- **Data Efficiency:** MTL can be particularly beneficial when dealing with limited data for each individual task. Sharing information across tasks allows the model to learn more effectively with less data.
- **Regularization:** MTL can act as a regularizer, preventing the model from overfitting to the specific features of any single task. (e.g Droput or early stopping).



QUESTIONS