# Deep Learning methods for pattern recognition

Ahmed Hammad

High Energy Accelerator Research Organization (KEK), Japan.

4th summer school at CTP, BUE

Fourth lecture

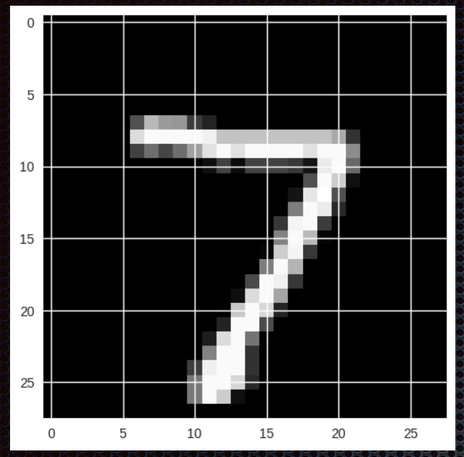
# Introduction

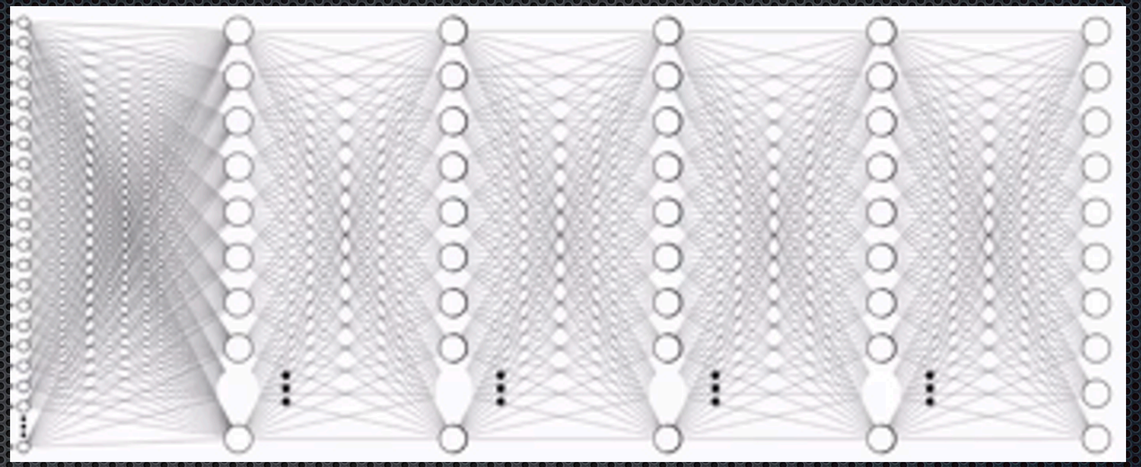
Why do we need CNN models ? DNNs can also analyze images !!

# Introduction

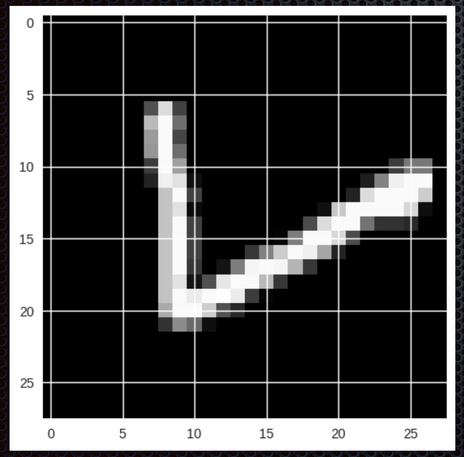
Trained DNN from lecture 3

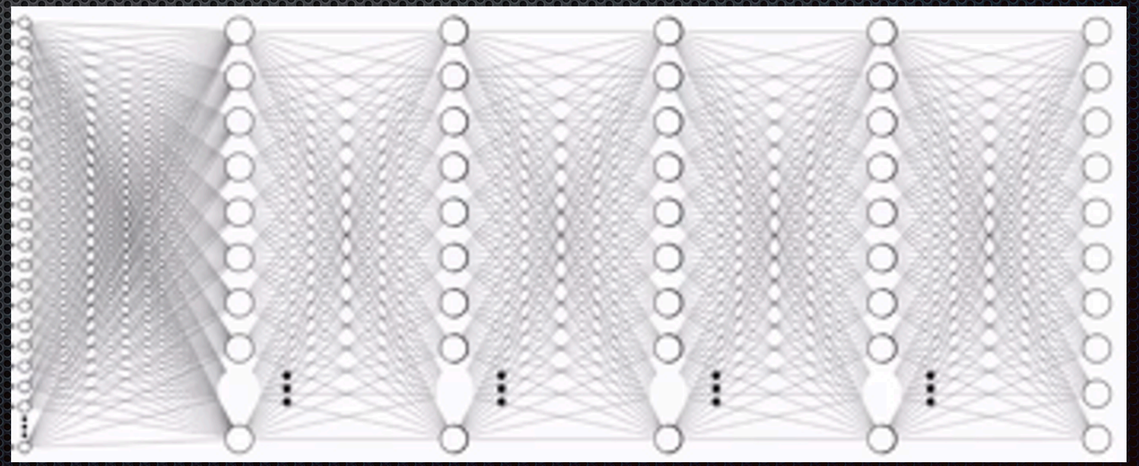
## Why do we need CNN models ? DNNs can also analyze images !!



1.5e-20
4.6e-16
1.5e-14
1.5e-15
3.4e-23
1.2e-22
4.2e-27
1.0e+00
1.2e-19
3.4e-09

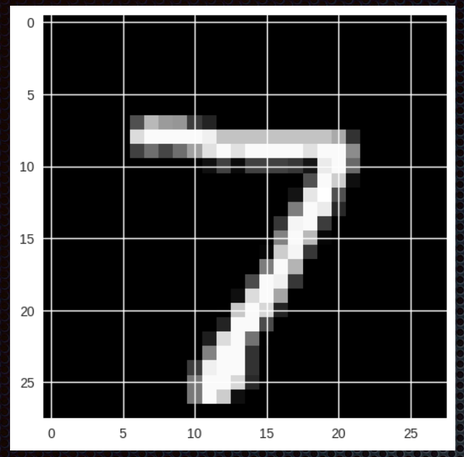


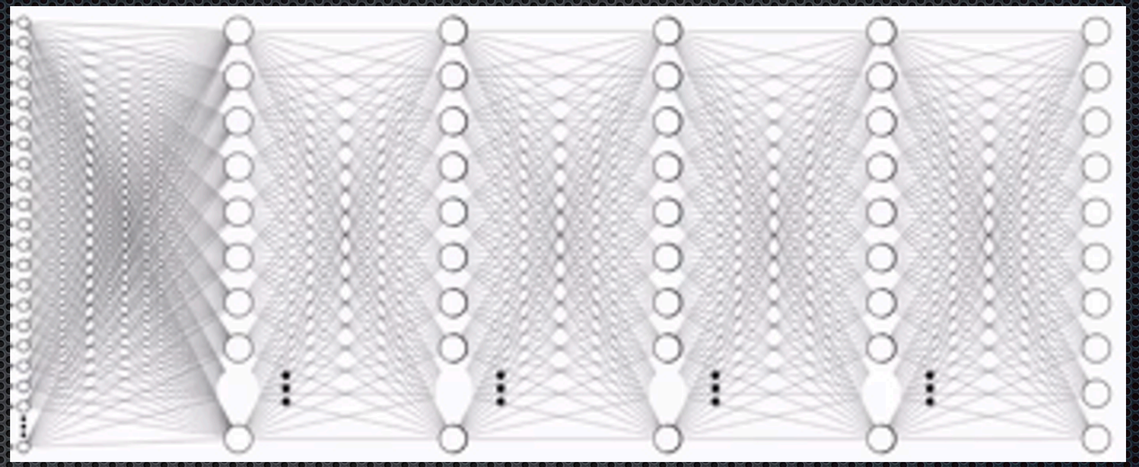



2.3e-04
1.8e-06
2.1e-04
4.3e-07
9.5e-06
1.0e-02
9.7e-01
1.2e-07
1.8e-02
1.3e-06

# Introduction

Why do we need CNN models ? DNNs can also analyze images !!



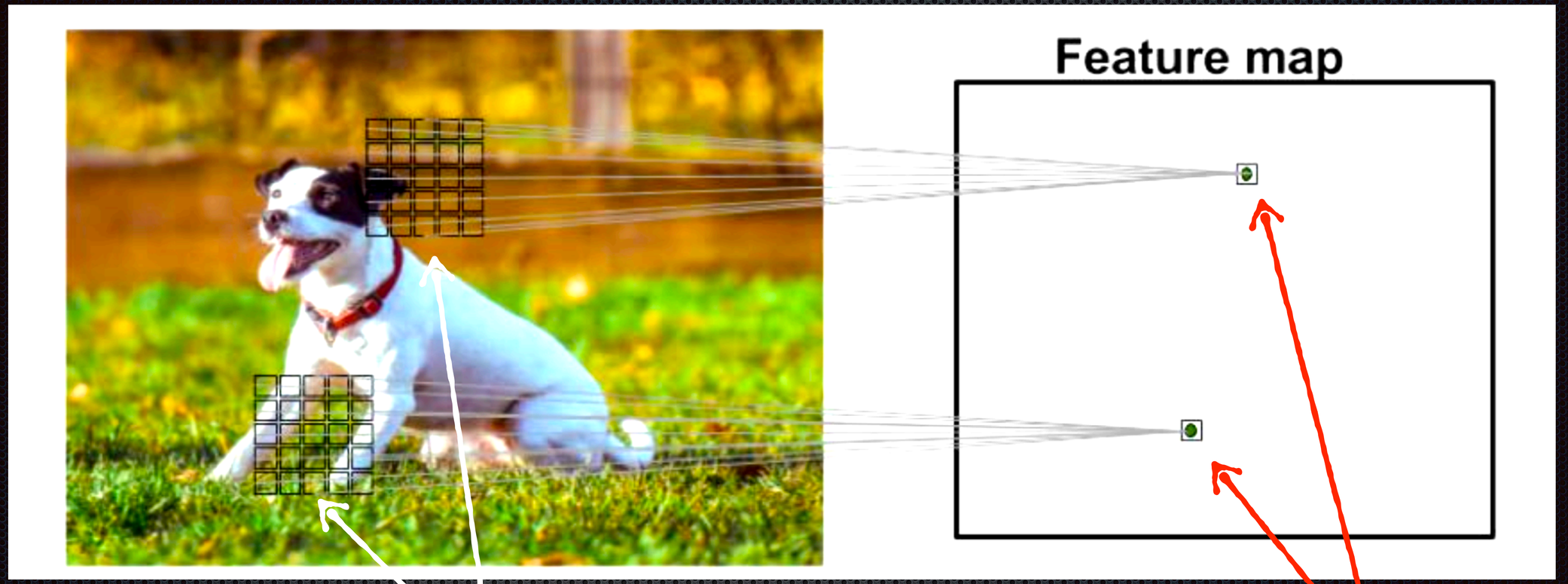
1.5e-20
4.6e-16
1.5e-14
1.5e-15
3.4e-23
1.2e-22
4.2e-27
1.0e+00
1.2e-19
3.4e-09

As a fully connected model

$$\text{Neuron Output} = \text{Activation function}\left(\sum_m \omega_m\, x_m + b\right)$$

The model learns the correlation between all pixels into the image and construct a global information about the image

# Introduction



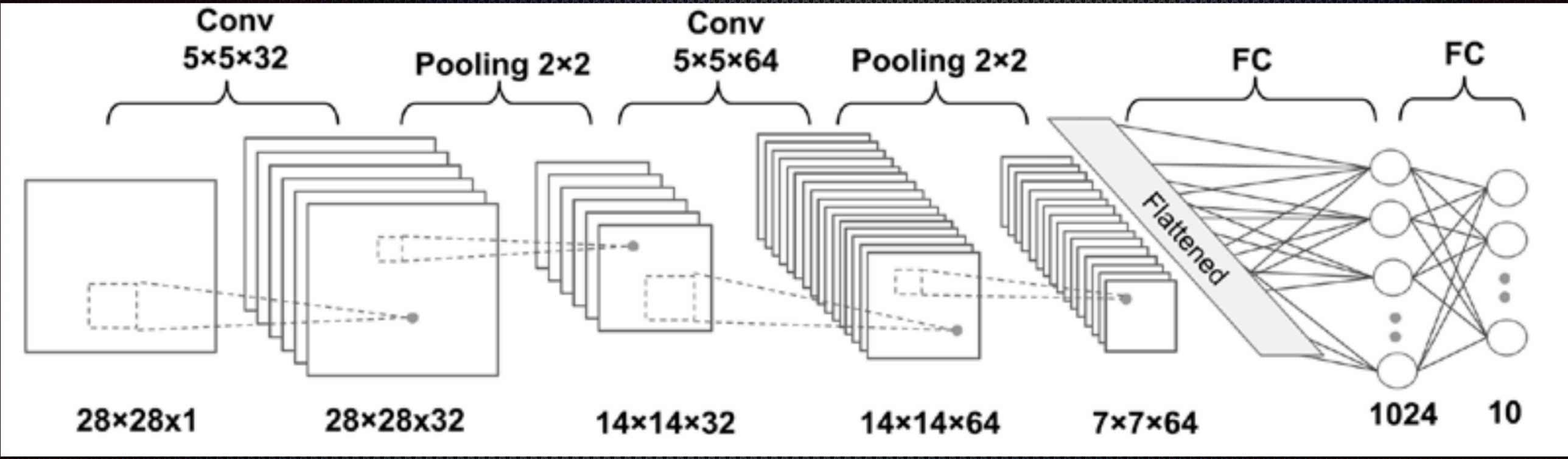

Filters to capture the local features of the images

Disconnected Patches

Filters share their weights internally but not with other filters

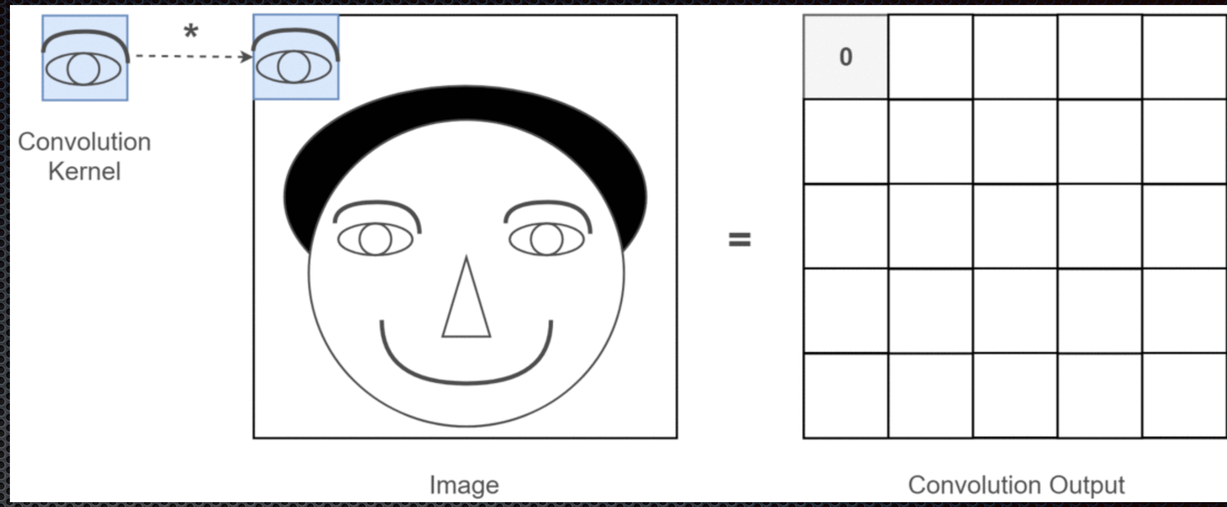
# Introduction

Example of deep CNN model

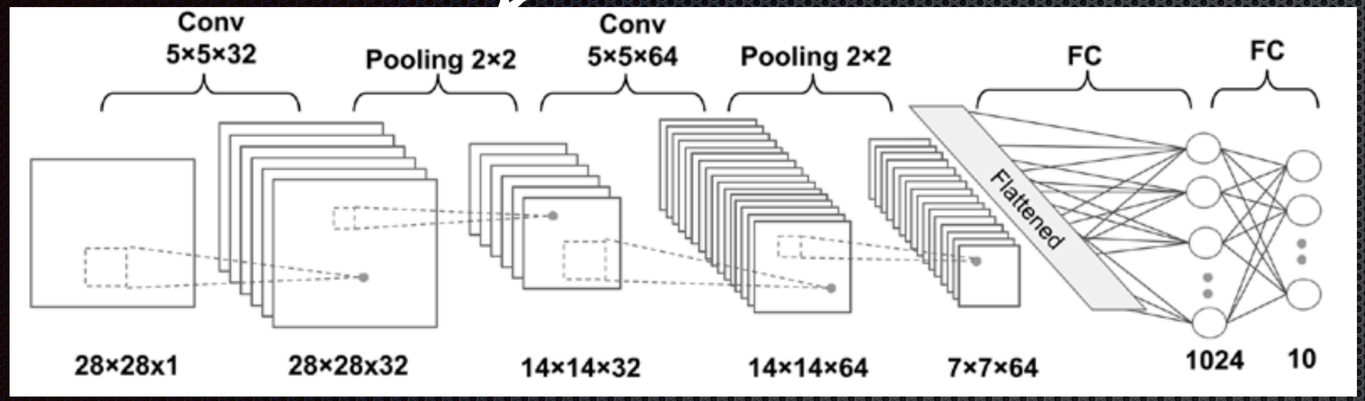



CNN = Convolution layers + Fully connected layers

# All together




Lower dimensions



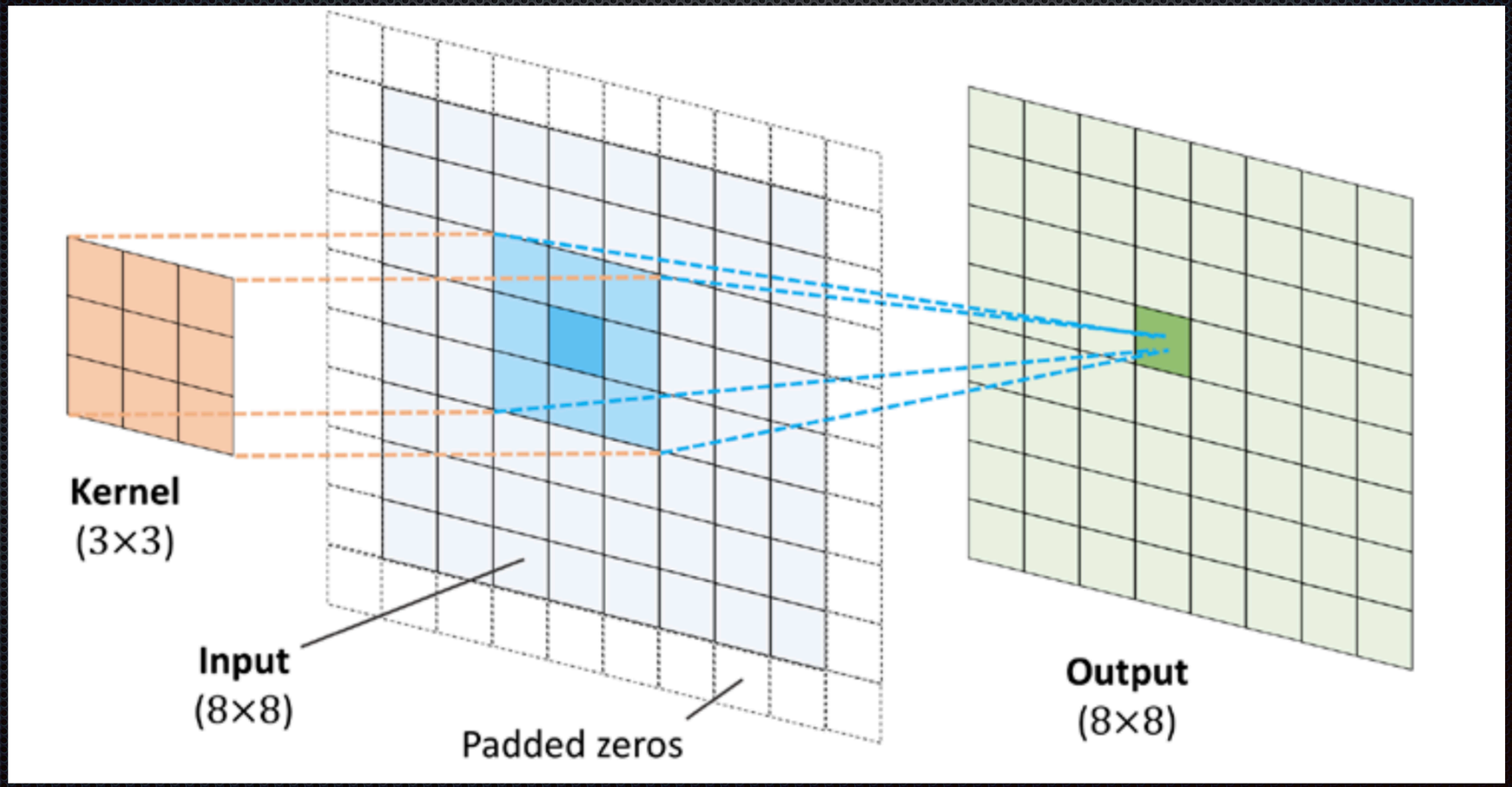

Using many filters to learn more local features, e.g. nose, eyes, etc

After the captured local information is mapped onto the decomposed latent space we use a fully connected layers to analyze these information and learn a global information about the input image

See lecture 4

# Convolution in two dimensions

$$y[i,j] = \sum_{k_1=0}^{k_1=m_1-1} \sum_{k_2=0}^{k_2=m_2-1} x_p[i+m_1-k_1, j+m_2-k_2]\omega[k_1,k_2]$$



# Convolution in two dimensions


| | | |
|---|---|---|
| 1 | 6 | 5 |
| 7 | 10 | 9 |
| 7 | 10 | 8 |

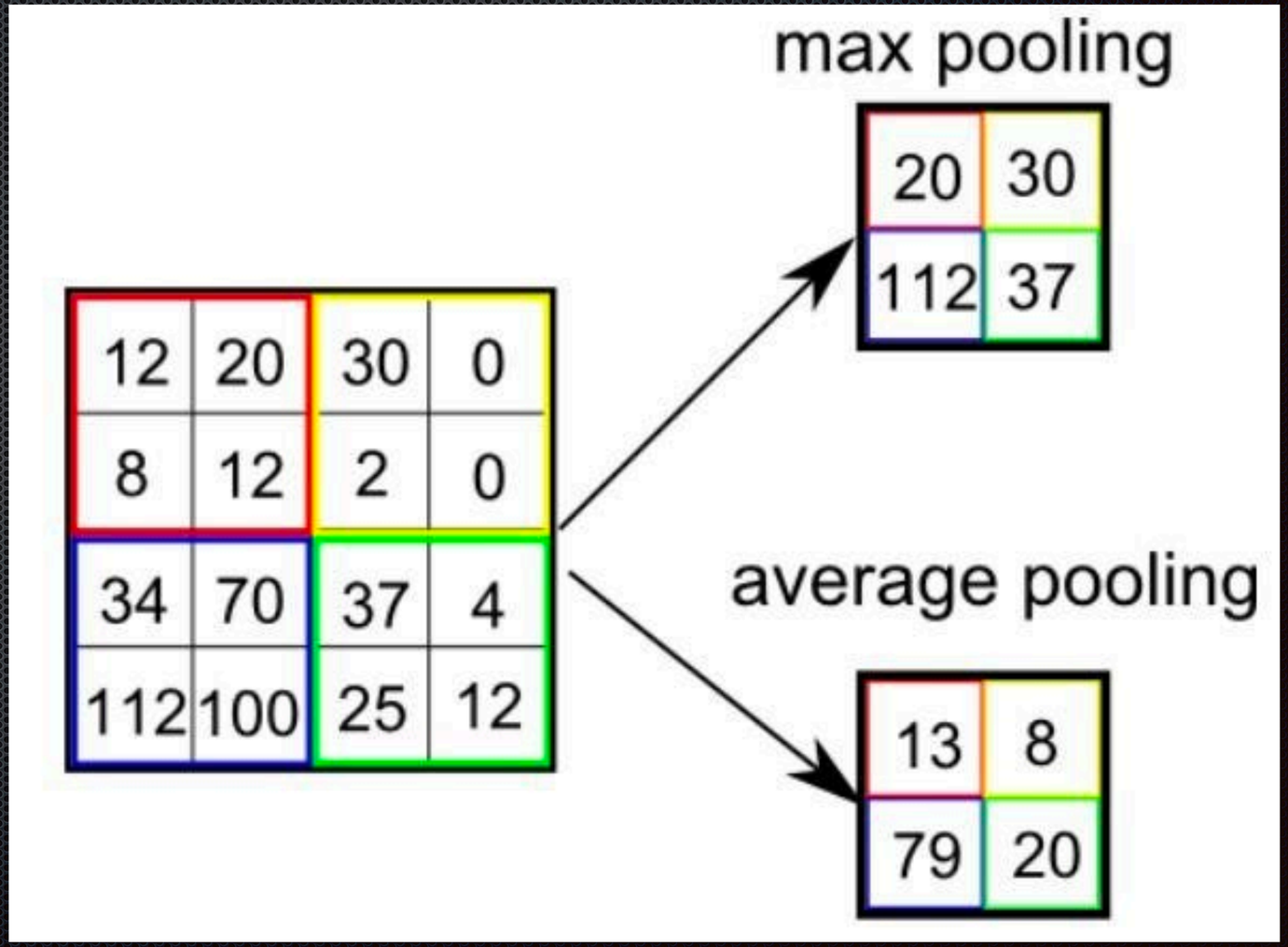
$$\text{Convolution output} = \frac{\text{Size of input vector} + 2*\text{Padding} - \text{filter size}}{\text{Stride}} + 1$$

$$\text{Convolution output} = \frac{(5,5) + 2*(1,1) - (3,3)}{(2,2)} + 1 = (3,3)$$

Local weights sharing

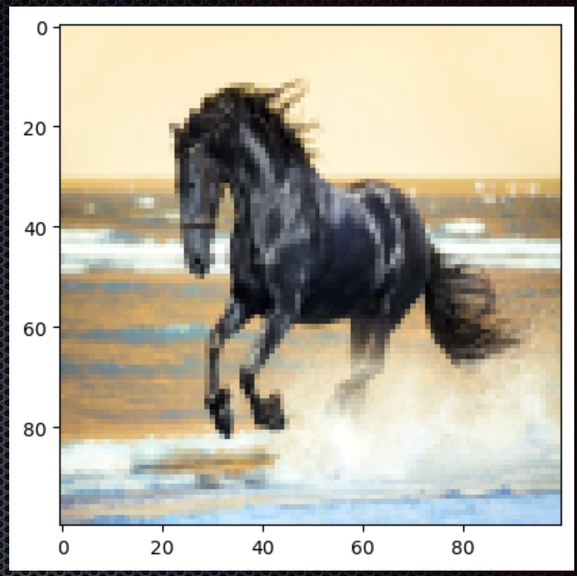
# Pooling

Pooling reduces the high dimensions Feature space to lower dimensions space (Latent space) to capture the characteristic information only



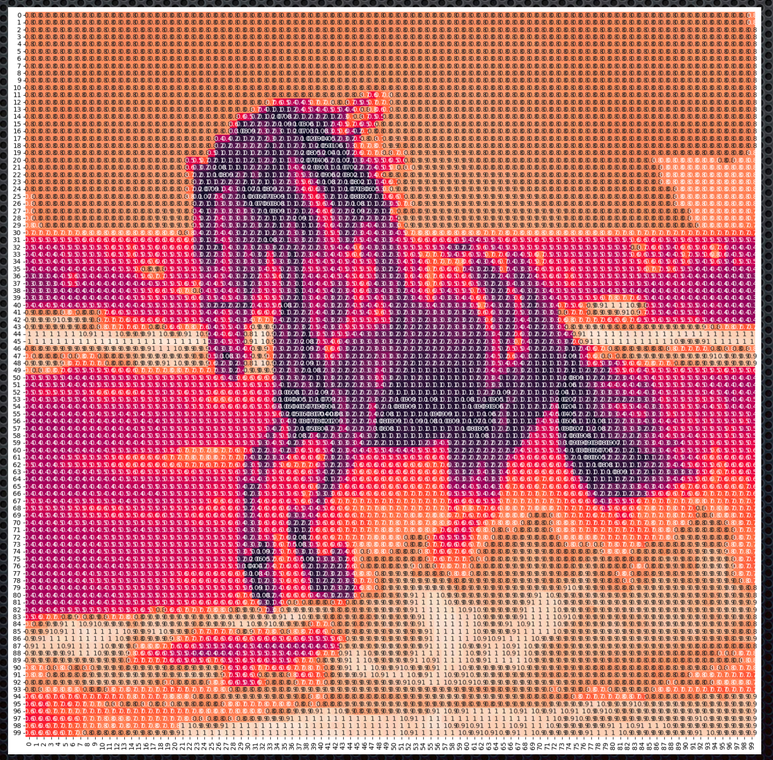
# Dealing with colored images

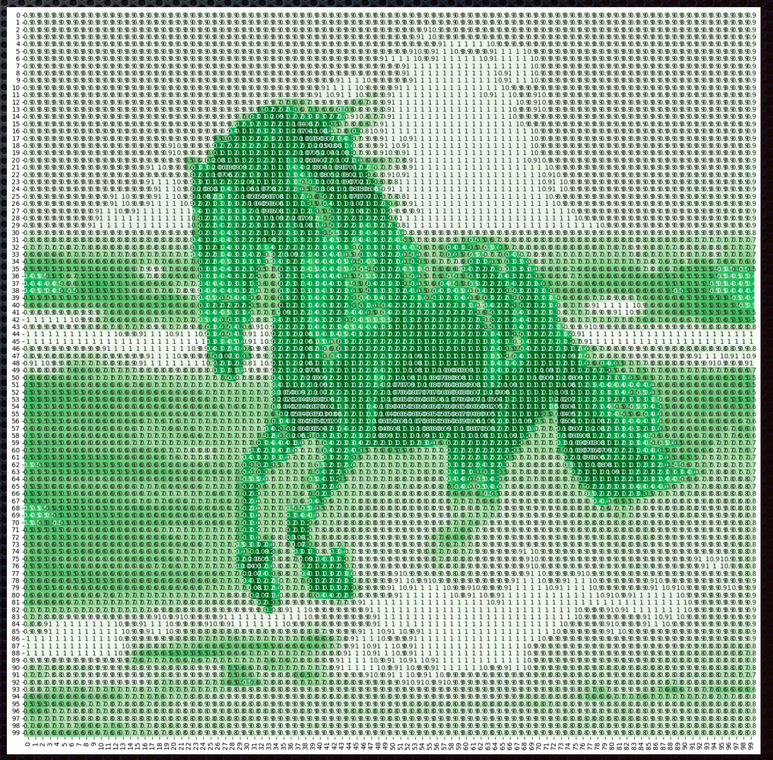
Original image
Dimension: (100,100,3)




Blue dimension
Dimension: (100,100,1)

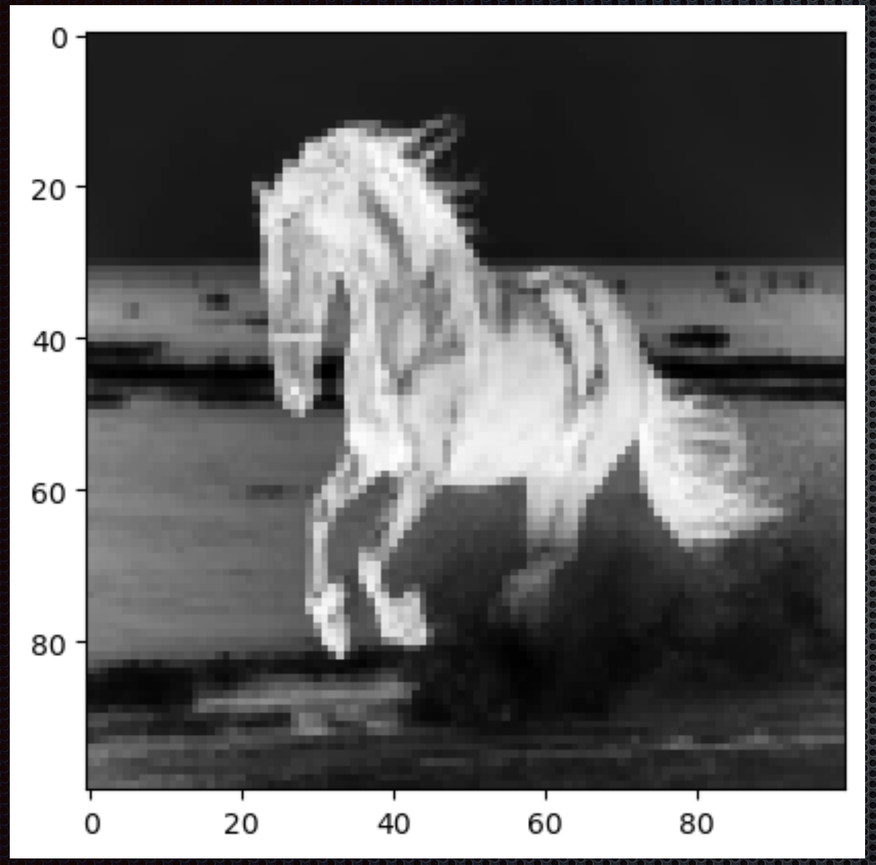
Red dimension
Dimension: (100,100,1)



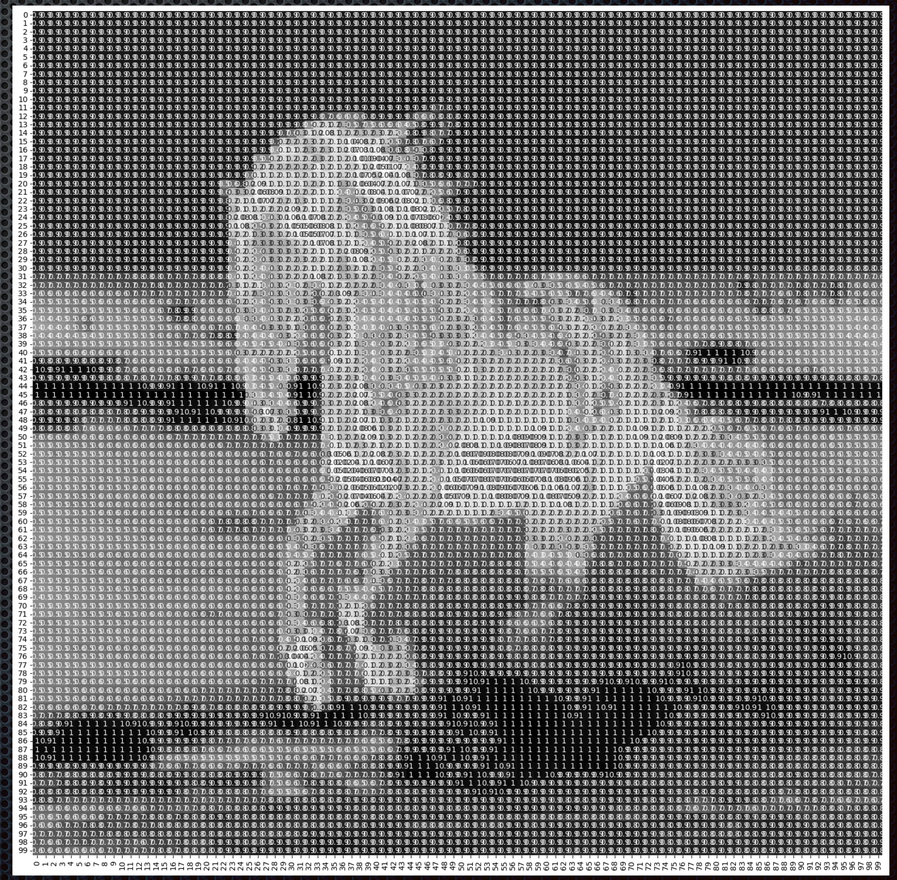


# Dealing with colored images

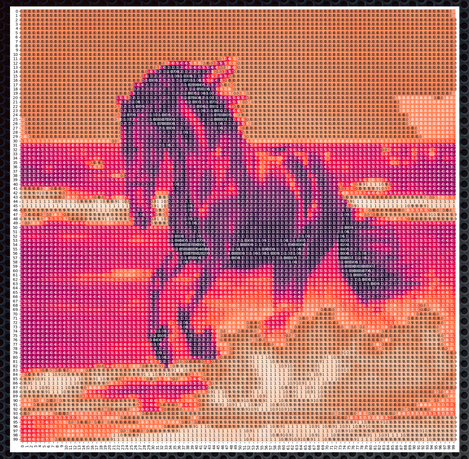
Original image
Dimension: (100,100,1)

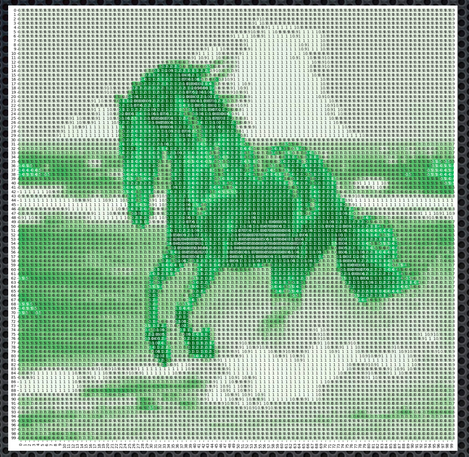


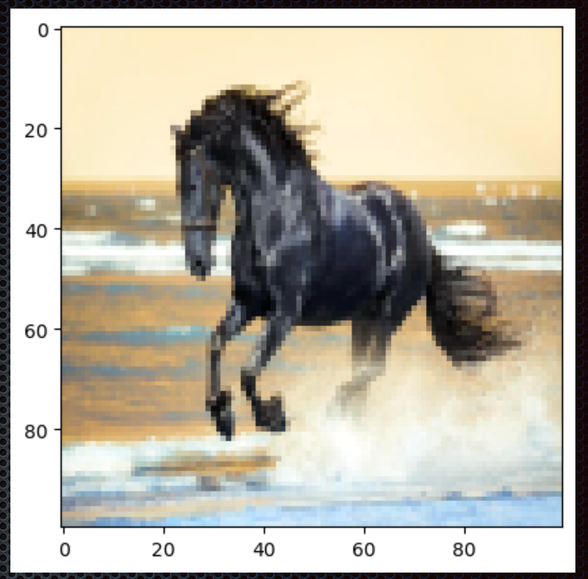

Only binary channel

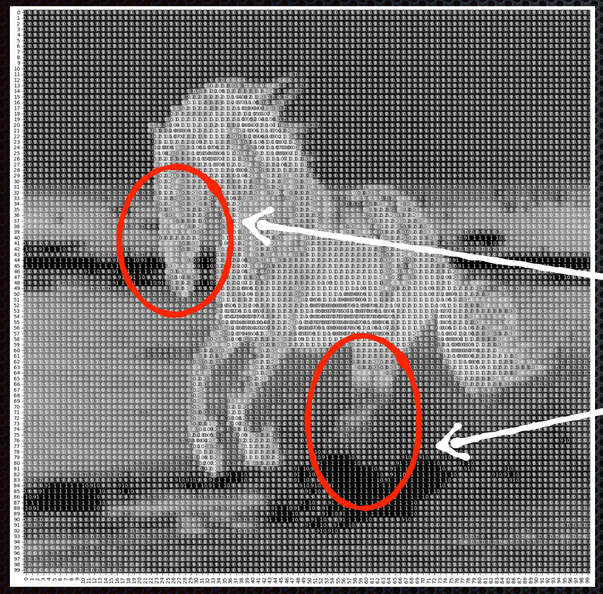


# Dealing with colored images







The image depth (color channels) can provide more local information than the binary image with depth 1

# Interpretable AI method

## Grad-CAM

# Grad-Cam

*Gradient weighted Class Activation Mapping (Grad-Cam) has been first Introduced in CNN model to visualize the most important pixels the model consider for his predictions.*

Arxiv:1610.02391

## *Gad-Cam works as the following:*

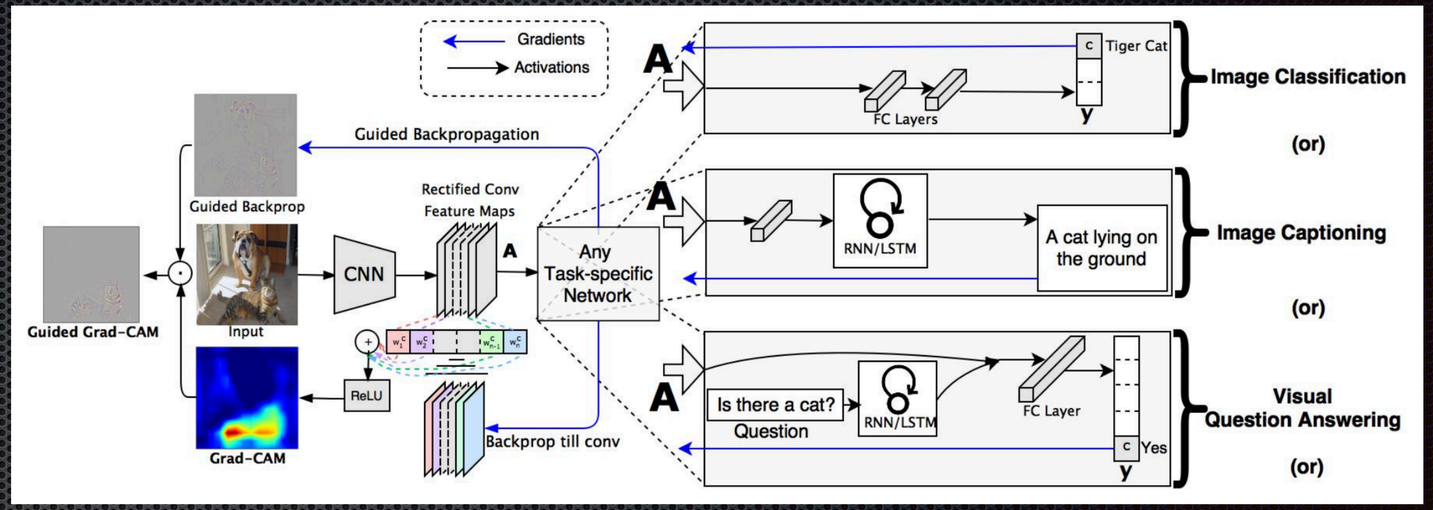
- *After training split the model from the last convolution layer.*
- *Compute the output of the last convolution layer (A)*
- *Compute the gradient of the class score of the second half of the model*
- *Compute the average of the gradients with resect to the spatial coordinates*

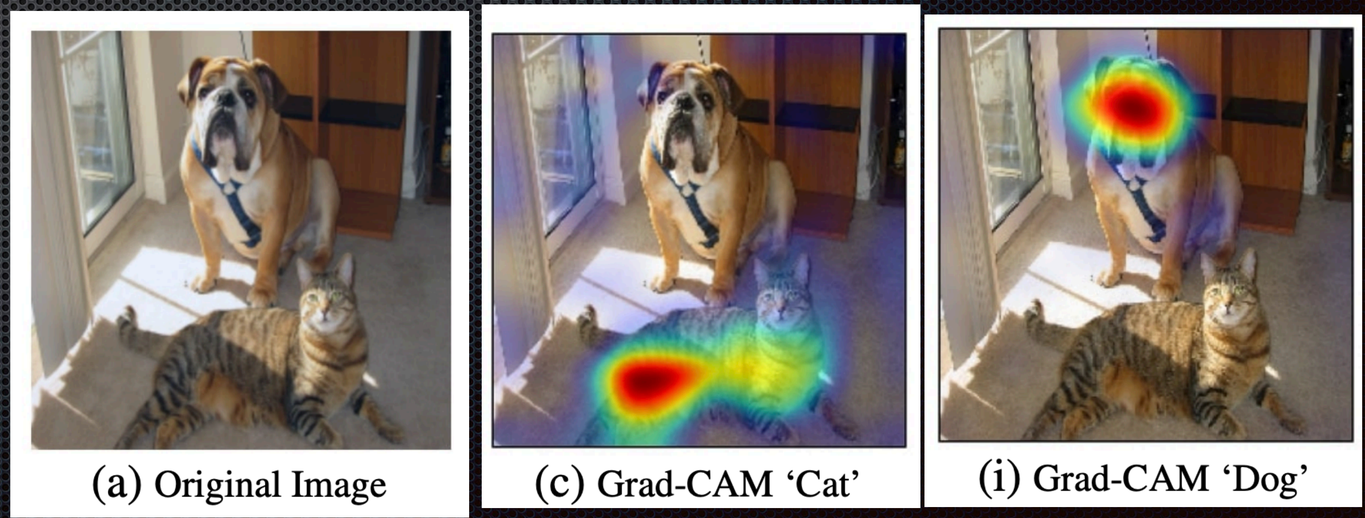
$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

- *Compute the weighted sum of the feature maps output (A)*

$$L^c_{\text{Grad-CAM}} = ReLU\left(\sum_k \alpha_k^c A^k\right)$$

- *The resulting heatmap indicates the spatial region in which the model focuses for predictions*



# Let’s go coding!!