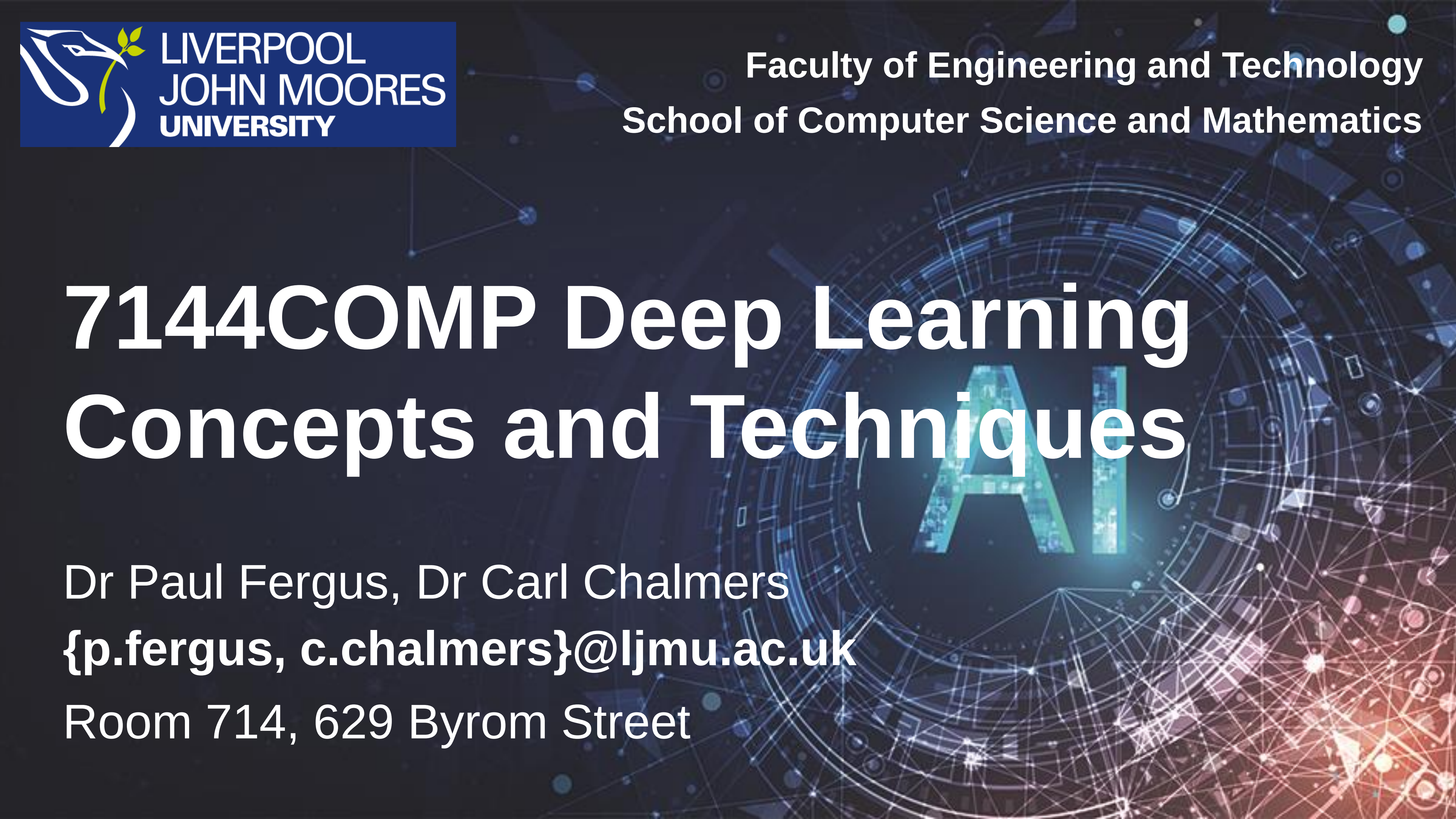


Faculty of Engineering and Technology

School of Computer Science and Mathematics

7144COMP Deep Learning Concepts and Techniques

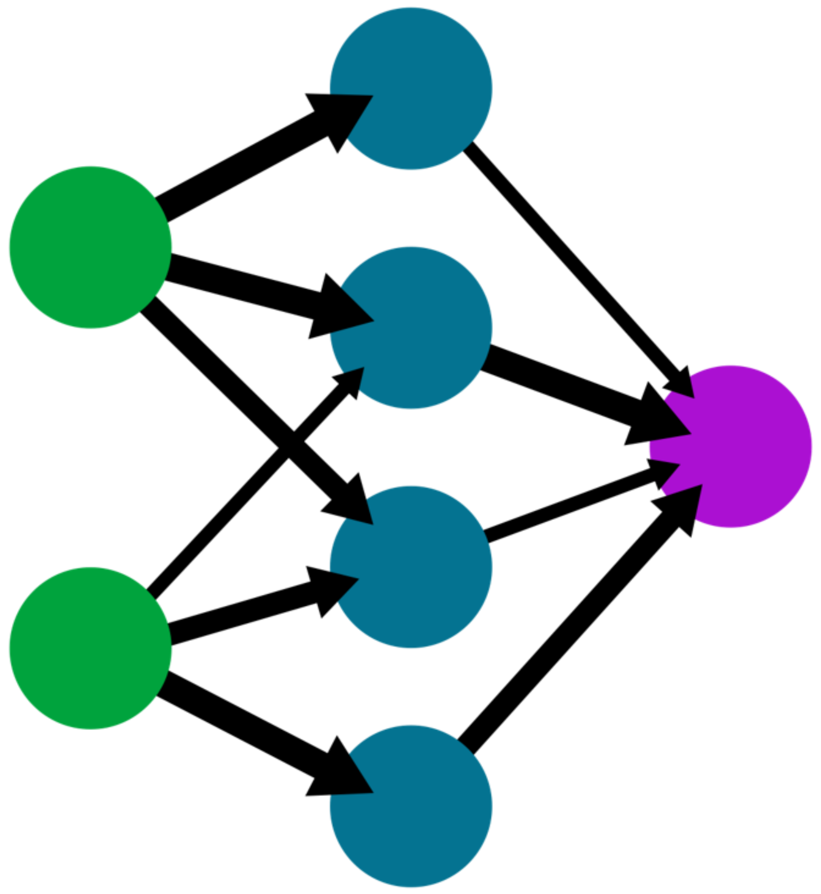
Dr Paul Fergus, Dr Carl Chalmers

{p.fergus, c.chalmers}@ljmu.ac.uk

Room 714, 629 Byrom Street

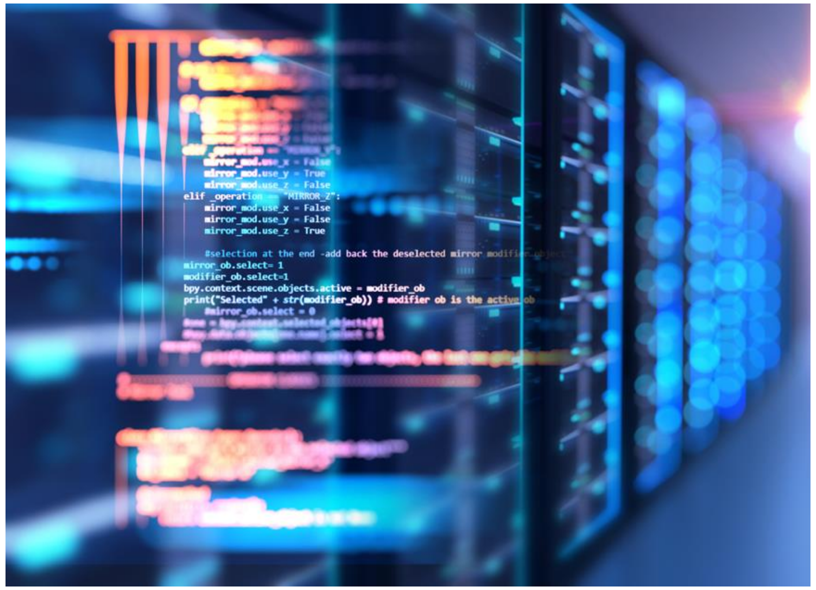
Lecture 5

Introduction to Convolutional Neural Networks (CNNS)



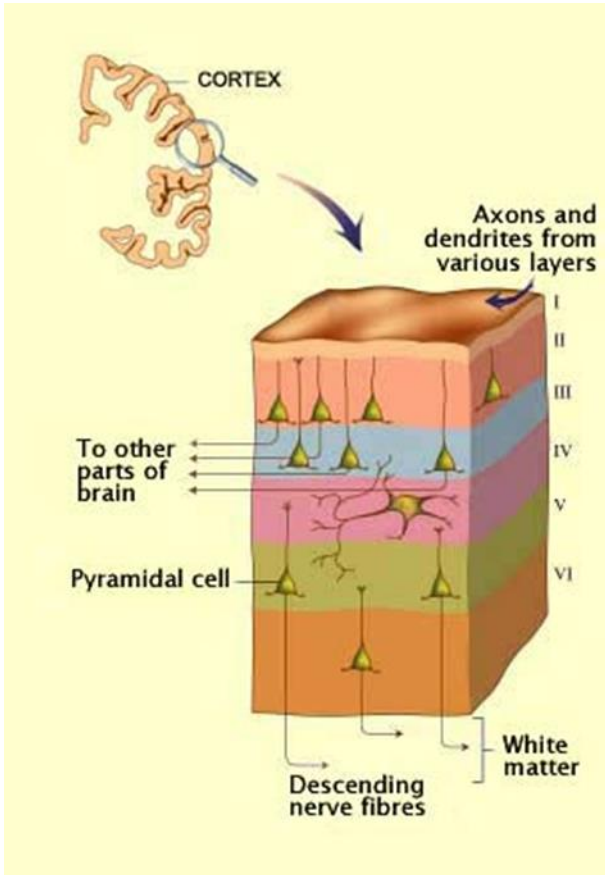
In this session…

- We will cover:
  - Introduction and biological comparison
  - Hubel and Wiesel
  - Image Data
  - Image Filters and Kernels
  - Convolutional Layers
  - Pooling Layers
  - Transfer Learning



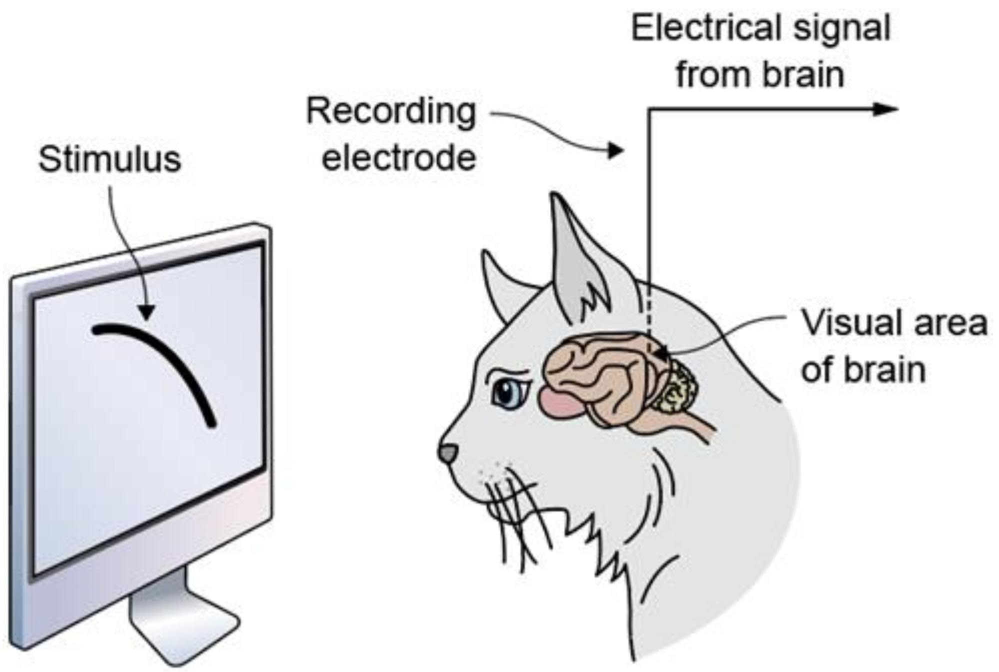
Convolutional Neural Networks (CNN)

- When your inputs are images or videos ANN have limited effect or use. Instead for this task we can use a type of network known as Convolutional Neural Network (CNN)
- CNNS are modelled on the primary visual cortex which is part of the cerebral cortex
- The primary visual cortex contains six layers which are responsible for extracting a variety of different features for example edges



Hubel and Wiesel

- The name Convolutional Neural Networks refers to a class of deep learning algorithms that were initially developed in the 1980s, inspired by the discoveries of Hubel and Wiesel
- In 1962, the two Nobel-prize neurologists Hubel and Wiesel noticed that single neurons in cats' neural cortex fired when specific regions of the visual field were stimulated



Convolutional Neural Networks (CNN)

- CNNS are extremely computationally expensive and have a variety of different architectures which comprise of filters, pooling and a final fully connected ANN
- The convolutional layers which are comprised of a number of filters along the pooling layers form the automatic feature extractor
- The fully connected layers (dense layers) form the classification

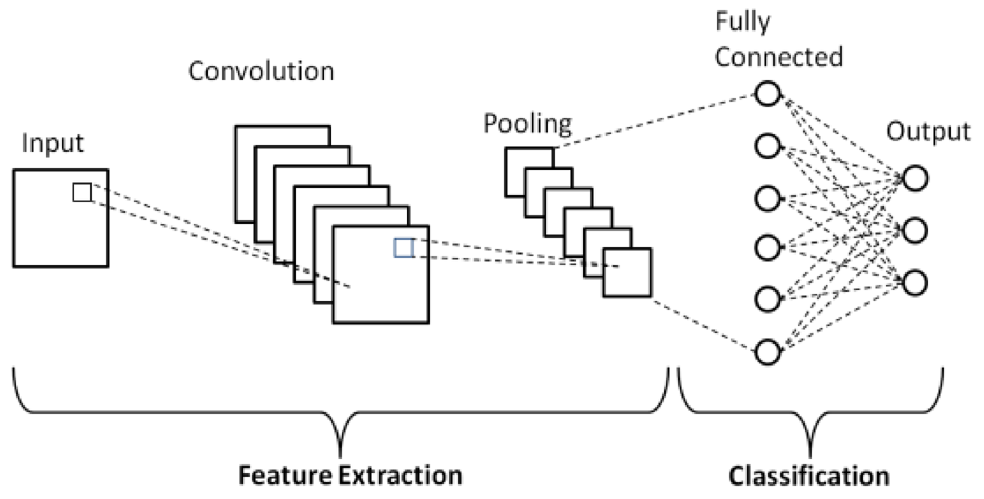


Image Filters and Kernels

- Filters are also known as a kernel which are a small matrix applied to an entire input space, such as an image
- Essentially kernels allow us to transform images. The image below shows an example of a grayscale image where the values in the matrix reside between –1 and 1 where –1 represents white, 1 represents black, and zero represents grey

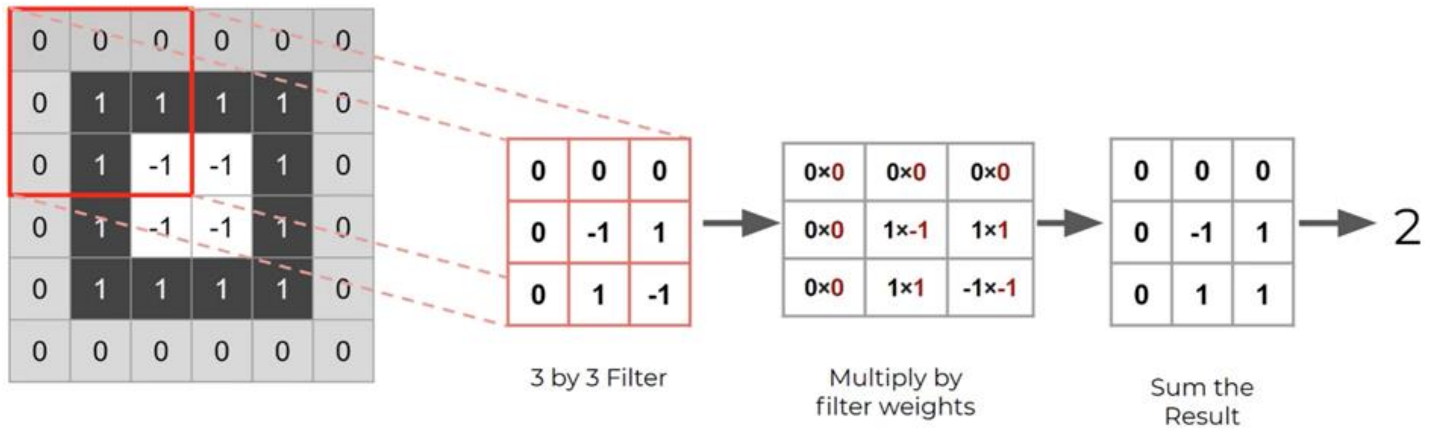


Image Filters and Kernels

- The size of the filter in this example is a 3 x 3 matrix which is applied to the input image staring at the top left
- As the filter moves across the image (stride) the pixel values are multiplied by the filter weights (these weights will be initialised and tuned during the learning process)
- In the final stage of the process the results are the summed to return the final value

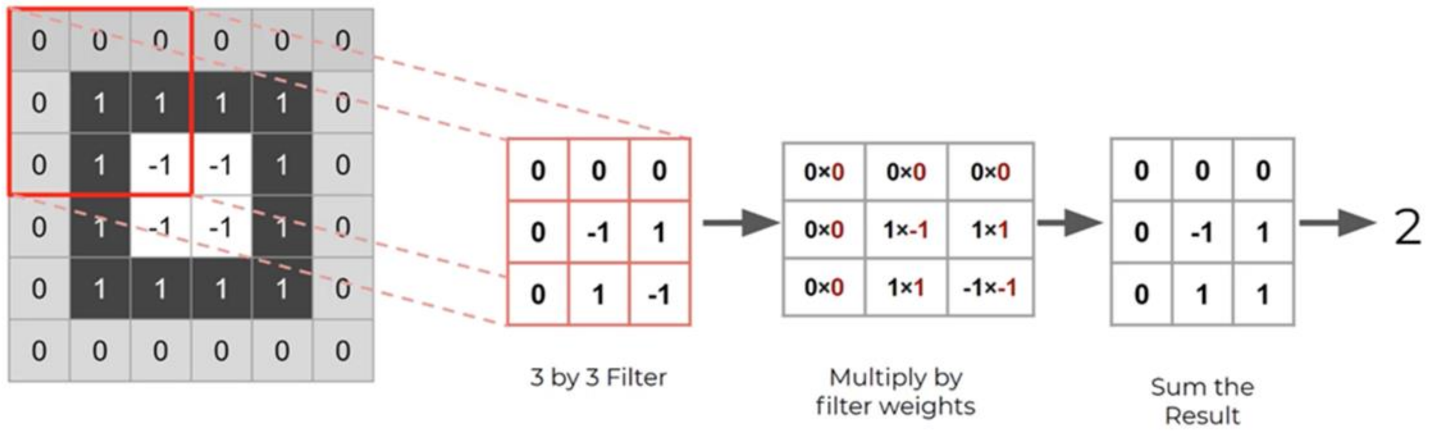
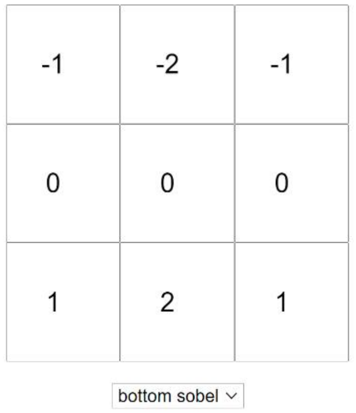


Image Filters and Kernels

- The stride length can be adjusted depending on the task. By default, the stride length must as least be 1 but can be increased depending on the problem
- There are a number of different image kernels which extract different features from a given image

Image Filters and Kernels

- For example, as shown below the Sobel operator is used in image processing and computer vision particularly in edge detection algorithms where it creates and image representing edges
- Bottom Sobel:



-1	-2	-1
0	0	0
1	2	1

bottom sobel

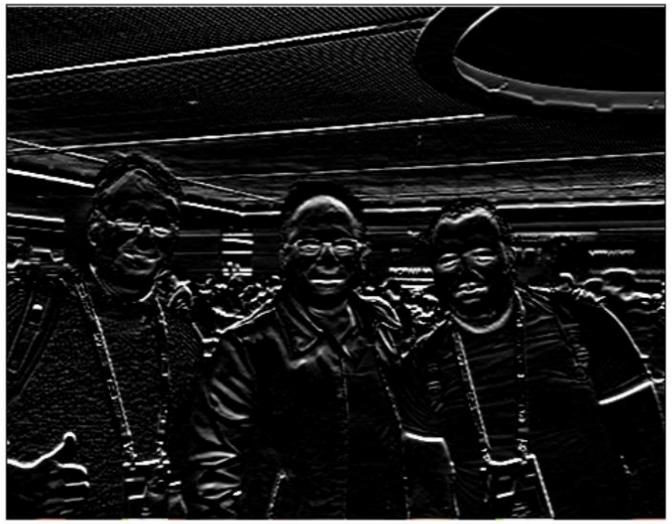
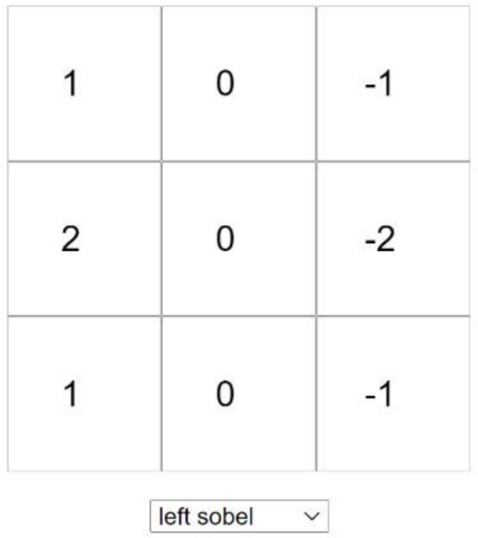


Image Filters and Kernels

- Left Sobel:



1	0	-1
2	0	-2
1	0	-1

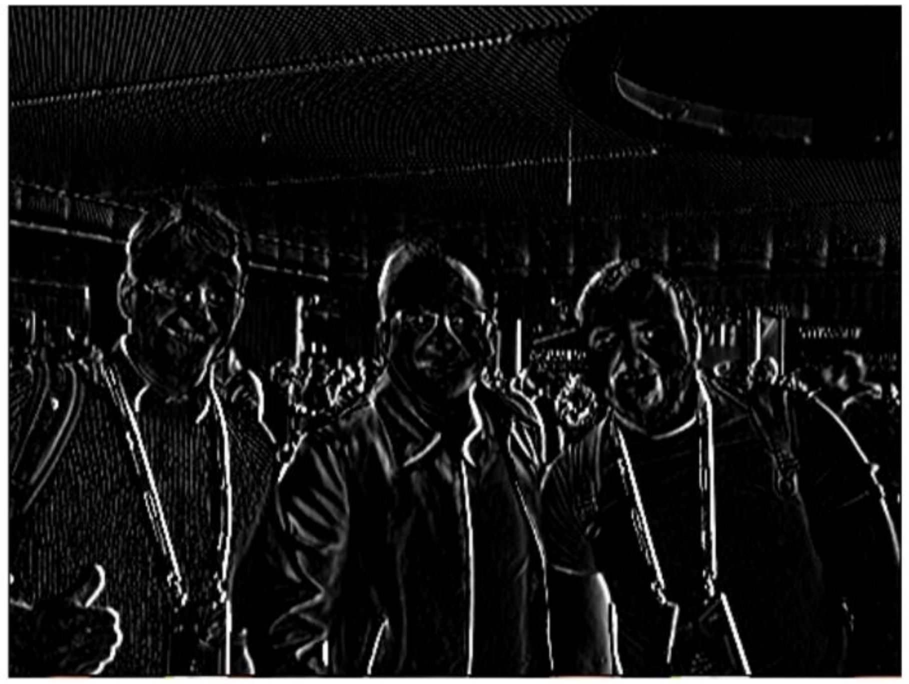
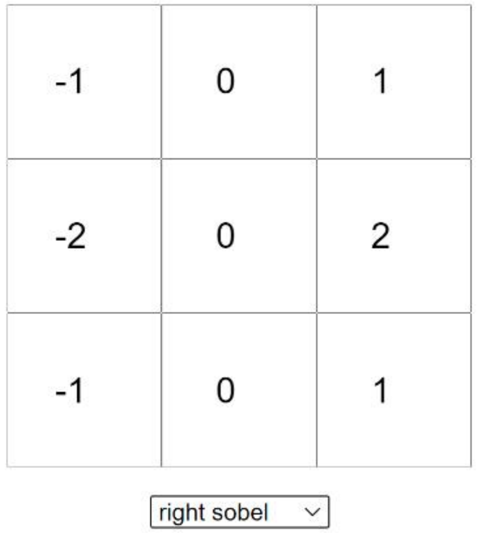


Image Filters and Kernels

- Right Sobel:



-1	0	1
-2	0	2
-1	0	1

right sobel

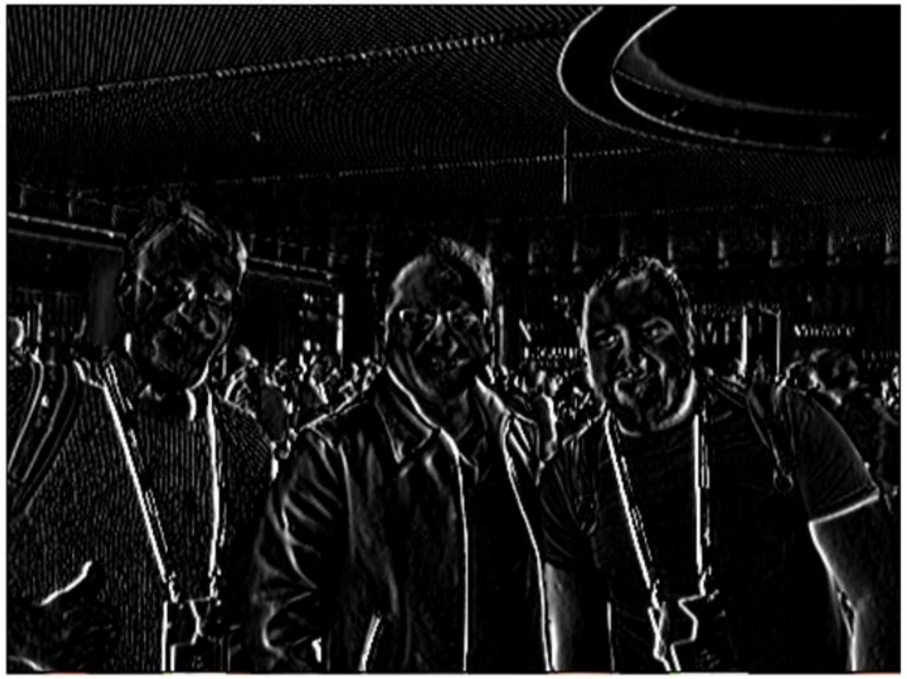
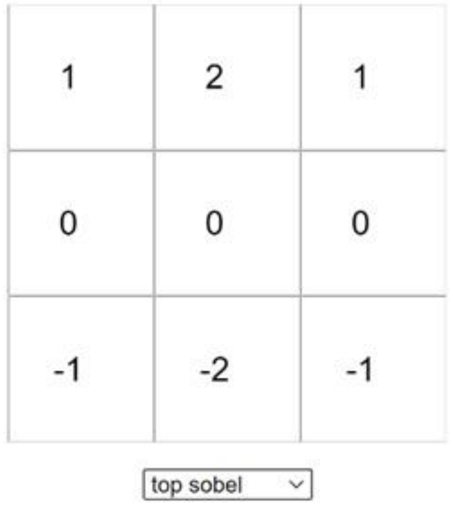


Image Filters and Kernels

- Top Sobel:



1	2	1
0	0	0
-1	-2	-1

top sobel

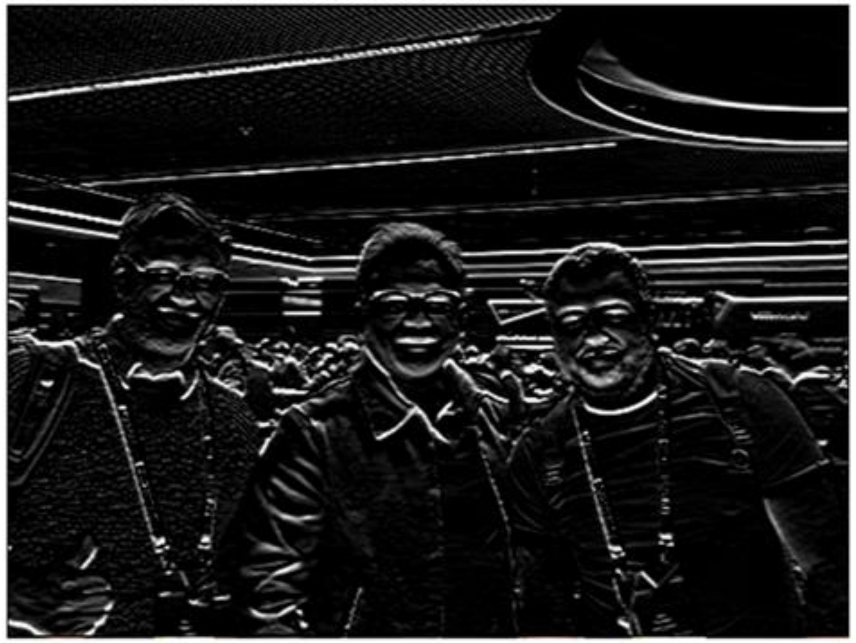
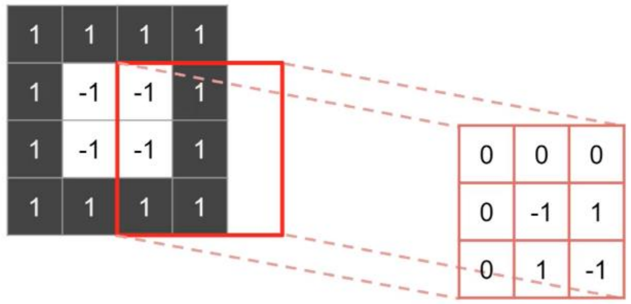


Image Filters and Kernels

- In the context of a CNN these features are referred to as convolutional kernels. The process of passing a kernel over an image is known as a convolution
- Kernels can overshoot (go beyond the edge of an image) which results in a loss of information which is compensated by using a technique known as padding. Here the edges of the image are padded with zeros which allows the entire image to be processed by the filters



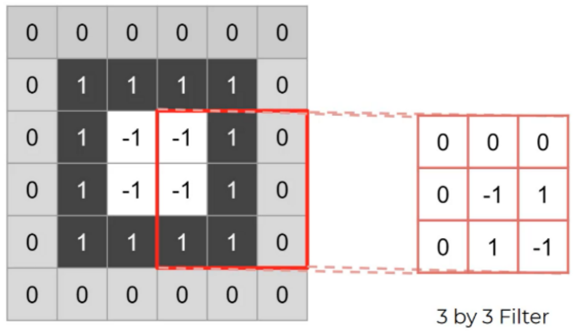
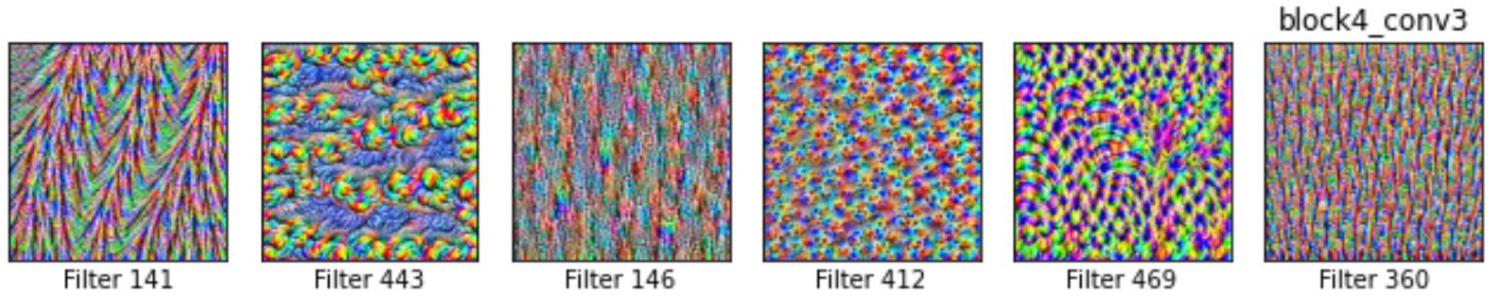
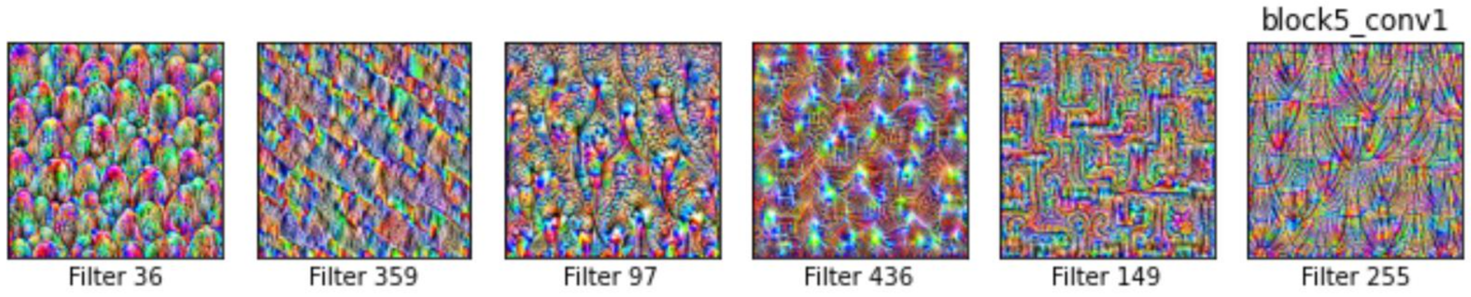


Image Filters and Kernels

Block 4 Conv3

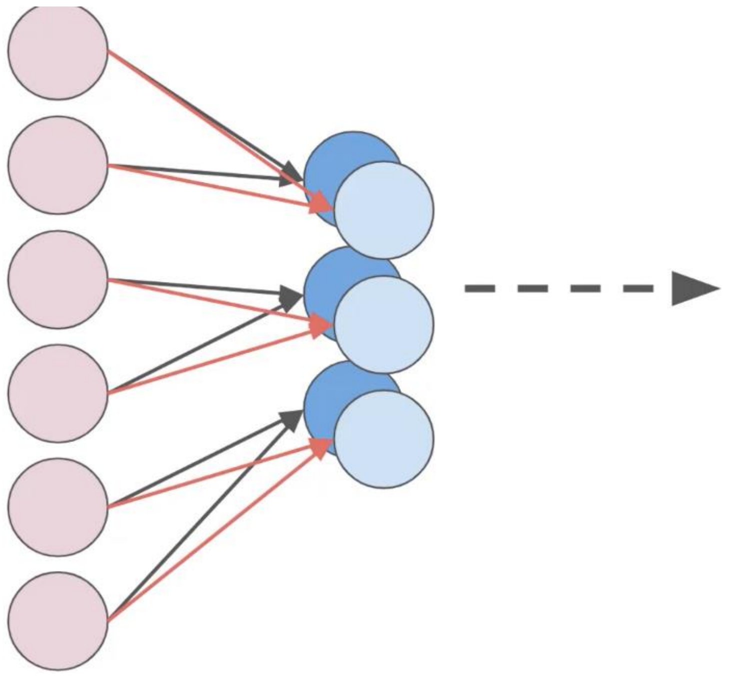


Block 5 Conv1



Convolutional Layers

- A convolutional layer when we apply multiple filters to the input image. A convolutional layer is trained to determine the best filter weight values for the input (learnable parameters)
- A CNN helps to reduce the number of parameters (which is an issue when processing images using fully connected MLP due to the large number of trainable parameters and increased computational requirements) by using local connectivity

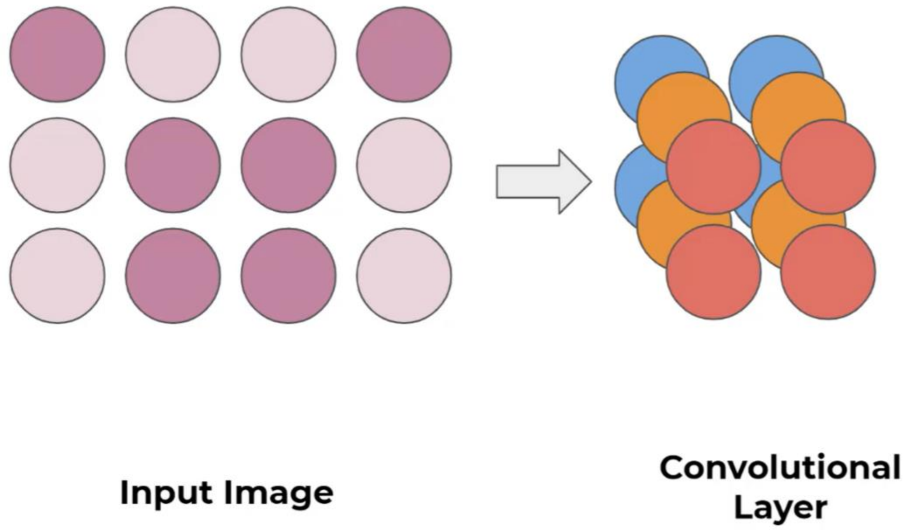


Convolutional Layers

- When configuring a CNN, you can specify the number of filters that you require
- There are common filters such as Sabel which are good for finding edges in an image whereby the weights are already known
- However, if you are working with say facial recognition you will unlikely to be able to manually determine the weights required to pick out features such as an eyebrow or nose
- Therefore, the CNN will determine a set of weights in order to extract the required features

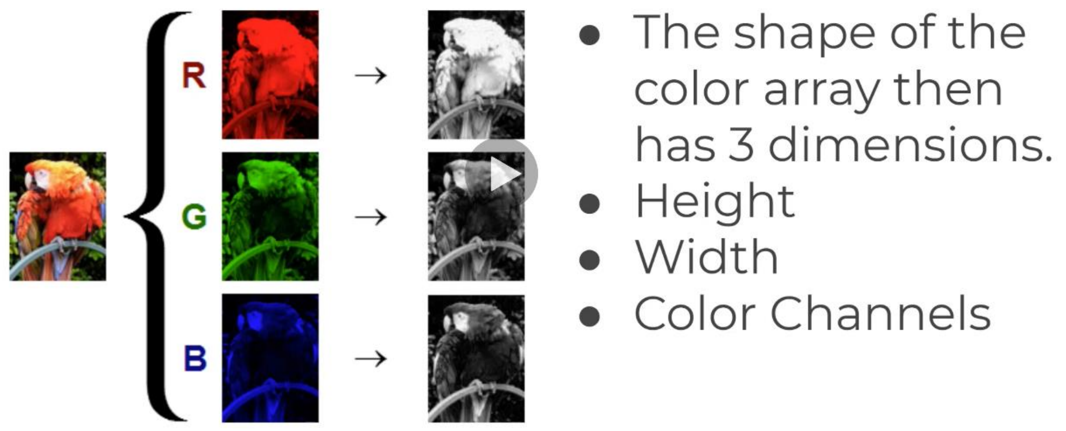
Convolutional Layers

- In the previous slides we looked at a 1DCNN however in basic images you would require a 2DCNN to process grayscale and preserve the 2D relational information in the convolutional layer
- Here the figure shows an example of a 2DCNN mapped to three filters which acts as a two by two matrix that has a stride of two



Convolutional Layers

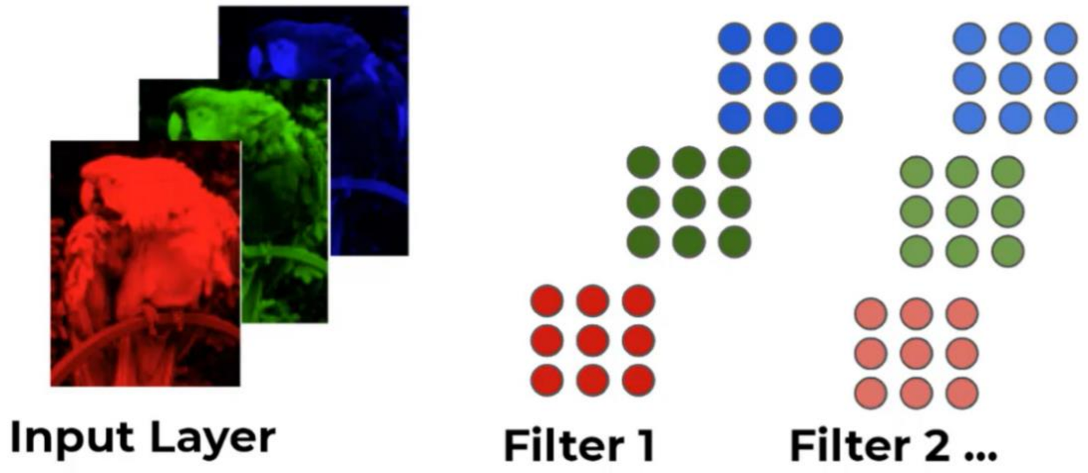
- Colour images can be thought as three-dimensional tensors which consists of red, green and blue (RGB) colour channels. Alongside the height and width, the dimensions of an RGB image would be HxWx3
- For example, when an image is imported with a resolution of 1280,720, inspecting its shape would result in a (1280,720,3) 3D matrix (tensor). Each of these values corresponds to 1280-pixel width, a 720-pixel height and a 2 colour channels



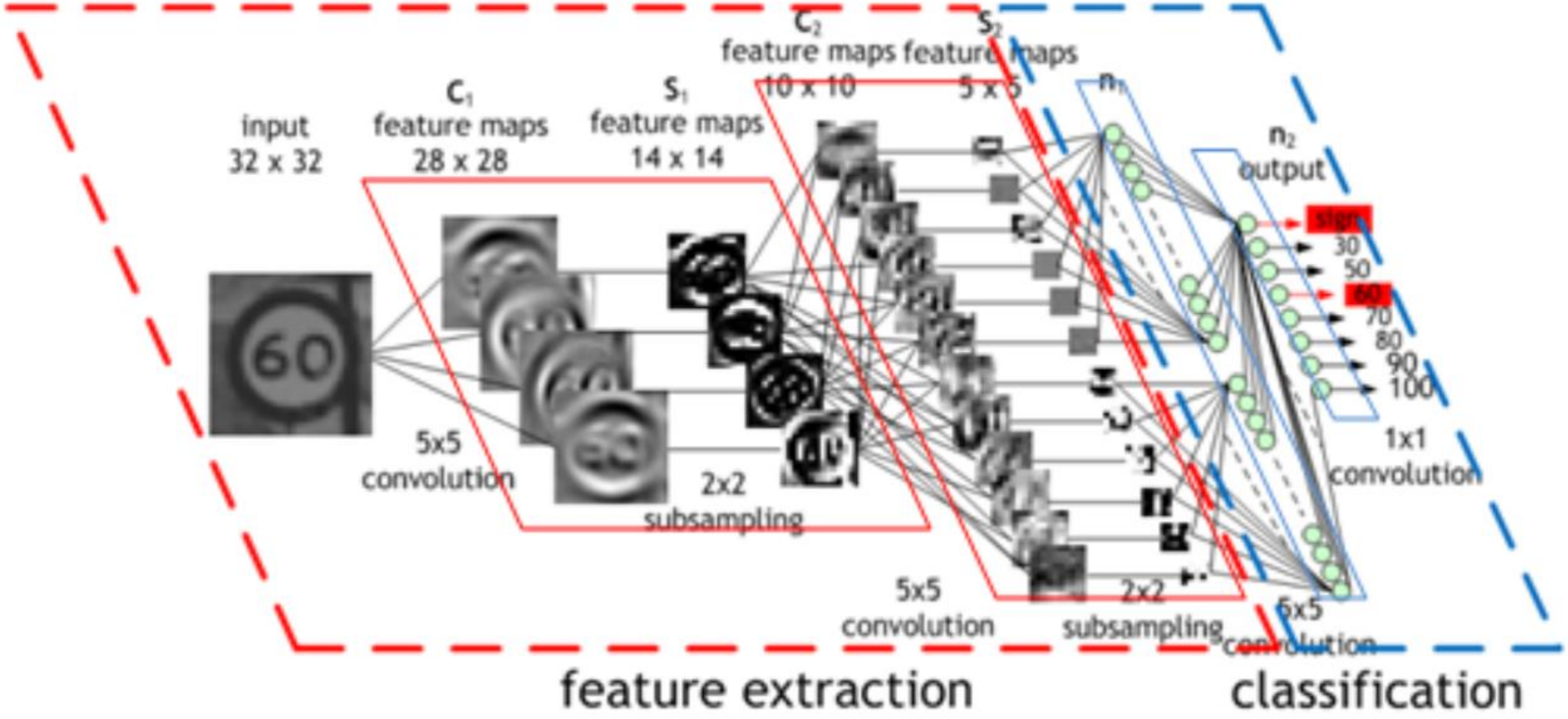
- The shape of the color array then has 3 dimensions.
- Height
- Width
- Color Channels

Convolutional Layers

- So, unlike the grayscale where a single channel of filters was used, in a 3D CNN we will require a set of features per channel (I.e. RGB). The figure below shows the layers in our input image and the layers in the corresponding filters. Just like you would consider a colour to be 3 dimensional you can think of your features as being 3 dimensional

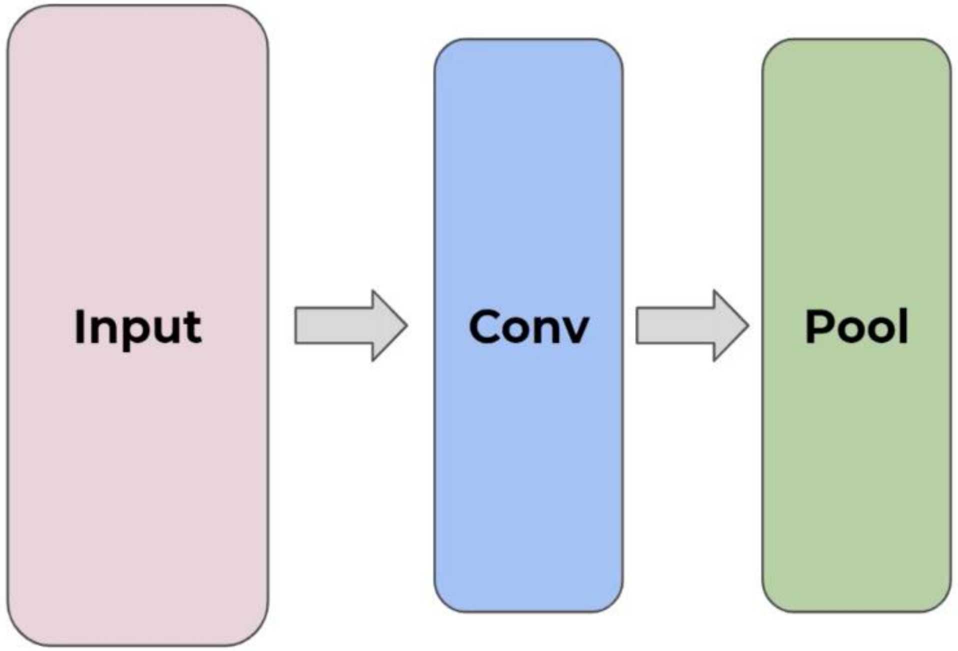


Convolutional Layers



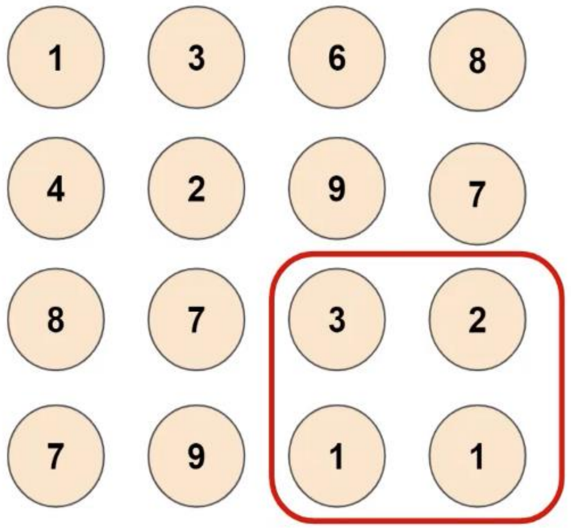
Pooling Layers

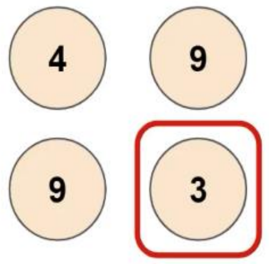
- Even though CNN’s use local connectivity dealing with large images, RGB and a variety of different features it can still result in many tuneable parameters. We can use pooling to reduce these parameters. Pooling layers accepts convolutional layers as input



Pooling Layers

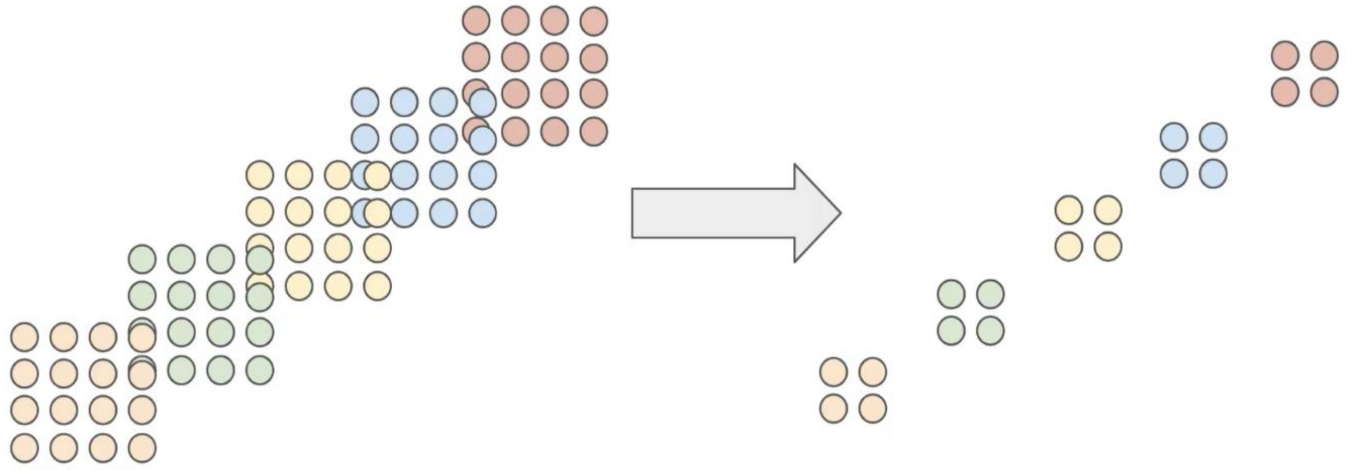
- A convolution layer will consist of a number of filters and we can use pooling to reduce its size of each filter. One common type of pooling is max pooling were the maximum value within a set stride for each filter in the convolutional layer is extracted





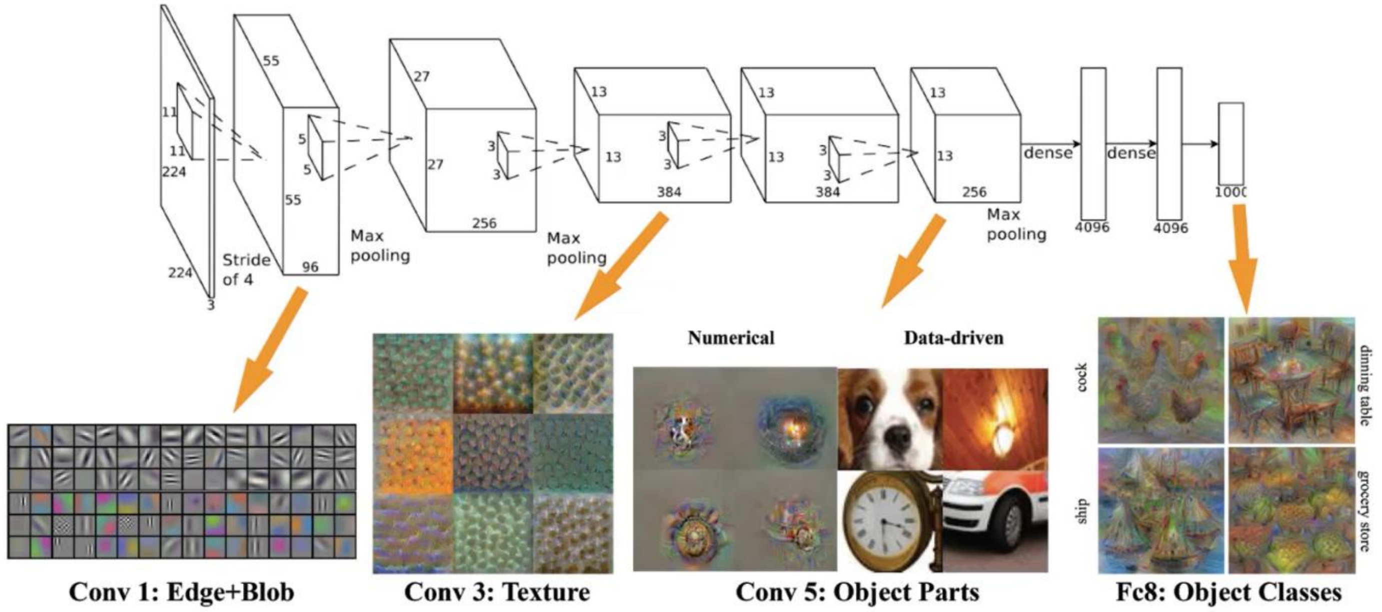
Pooling Layers

- While the previous slide showed one matrix resulting from one filter in the convolutional layer you will need to do this for each filter implemented in your solution. Note that pooling is performed on each filter in turn and not on a cross section of the combined filters. Even a small pooling kernel of 2 x 2 with a stride of 2 will remove 75% of the input da



Example (AlexNet)

- Below shows a basic CNN called AlexNet which competed in the ImageNet large scale visual recognition challenge in 2012 and was widely considered to be the first CNN in the challenge. Here the diagram shows the detailed interactions between the convolutional layers, max pooling layers and the fully connected dense layers



Transfer Learning

- Training CNNS form scratch which involves tuning the filters in the convolutional layers requires a significant number of images and computational resources
- There are a variety of different datasets which have been used to train a CNN from scratch
- The most common dataset used in training CNNS is the COCO dataset which contains 200,000 images with 1.5 million object instances and 80 object categories
- Beyond your tech giants very few people have the required infrastructure (and data) to successfully train a CNN from scratch

Transfer Learning

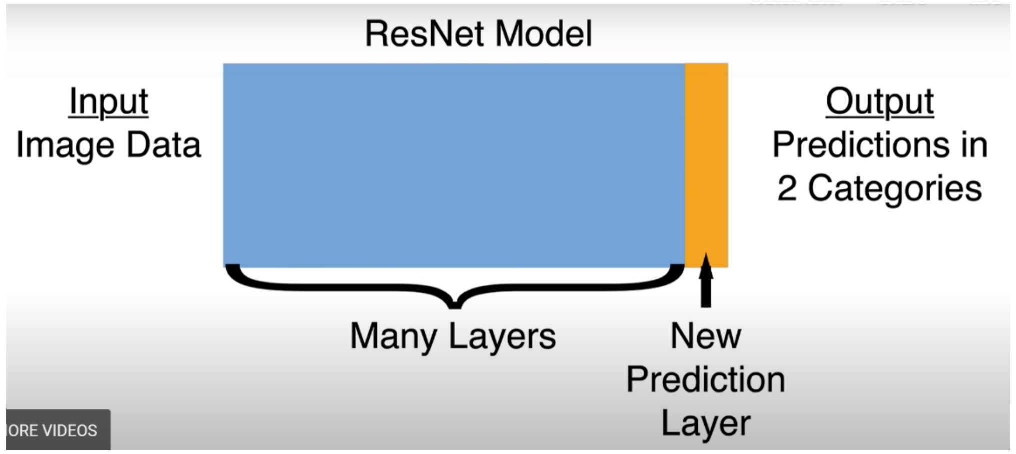
- Luckily a number of the big tech organisations offer pre-trained models which you can use to make your own predications as long as the images of interest are contained in the original model
- If you want a model to detect new objects which are not contained in the original model you would have to extend these models to include new objects of interest
- This is achieved using a technique known as transfer learning
- For example, if want a model to detect that’s not in the original model such as a giraffe and rhino you could train a new CNN from scratch to detect these new objects however to get results we would need to train the model on 1000’s of images

Transfer Learning

- The other way is to use transfer learning to train a new model but with far less data
- Transfer learning takes what a model has learned previously and applies to a new domain. The early layers of a CNN identify simple shapes while the later layers learn more complex shapes while the very last layer makes the predications
- Most layers of a pre-trained model are useful in new applications because most visual problems involve similar now level visual patterns. So, we can reuse most of the pre-trained model and replace the final layer that was used to make predictions

Transfer Learning

- Transfer learning is of particular interest for industry as we can build upon previously acquired knowledge without the need to start from scratch
- As we begin to build these models over time much more advance learning and capabilities will emerge



Next Session

- Image Classification and Object Detection Part 1
  - Computer vision overview
  - Historical context
  - Why computer vision is useful
  - Data-Driven Approach (Collect a dataset of images and labels)
  - Famous datasets
  - Hardware Accelerated Deep Learning (CPU vs GPU)
  - Training and Associated Hardware (Development Systems, Training Systems, Inferencing Systems)
  - Tensor Processing Unit (TPU)
  - Other Hardware Considerations
  - Distributed Training
  - Model Parallelism