



ANNs vs CNNs

Week 1: Computer Vision



Agenda

- Image classification using ANNs
- Challenges with ANNs
- CNNs over ANNs




Image classification using ANNs

Image classification using ANNs

Let's say we want to build a neural network using ANNs to classify an input image as a cat or a dog. For this, we'll be using a set of images of cats and dogs.

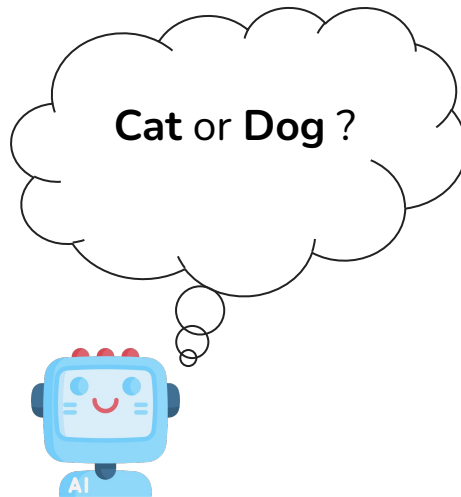


Image classification using ANNs

- If the images are RGB, we know that each image will have 3 channels, which will be represented by 3 matrices - each matrix numerically representing the pixel intensity values from that channel.
- The information about brightness, contrast, edges, shape, texture, shadows etc. of an image **does not depend on color**. So the RGB representation of images only adds to the complexity, and may be replaced by their grayscale version instead.

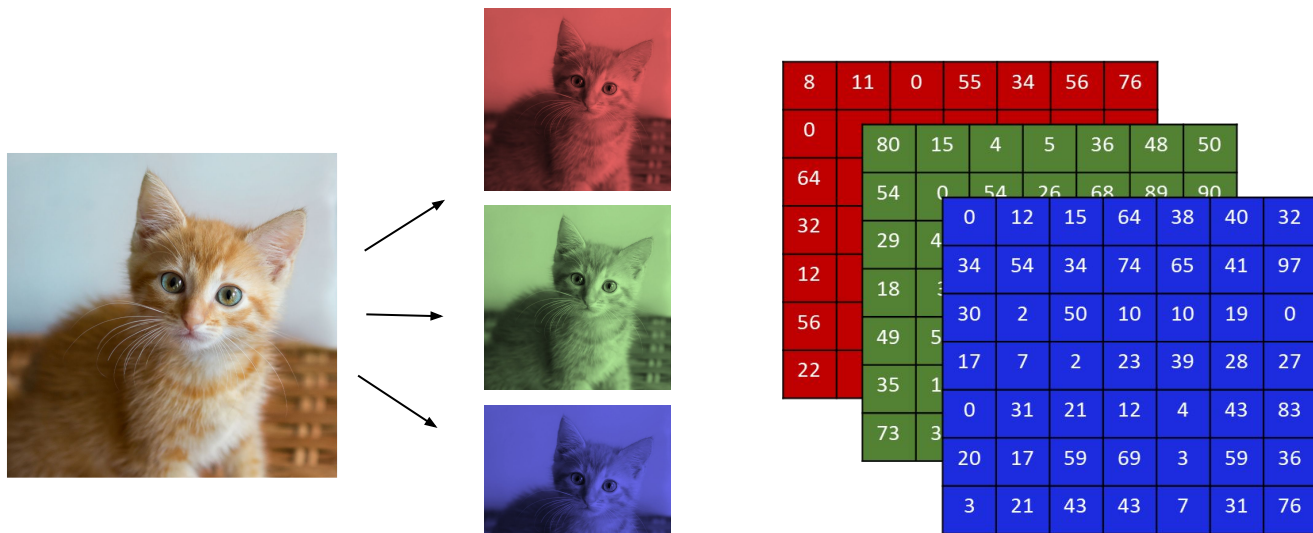
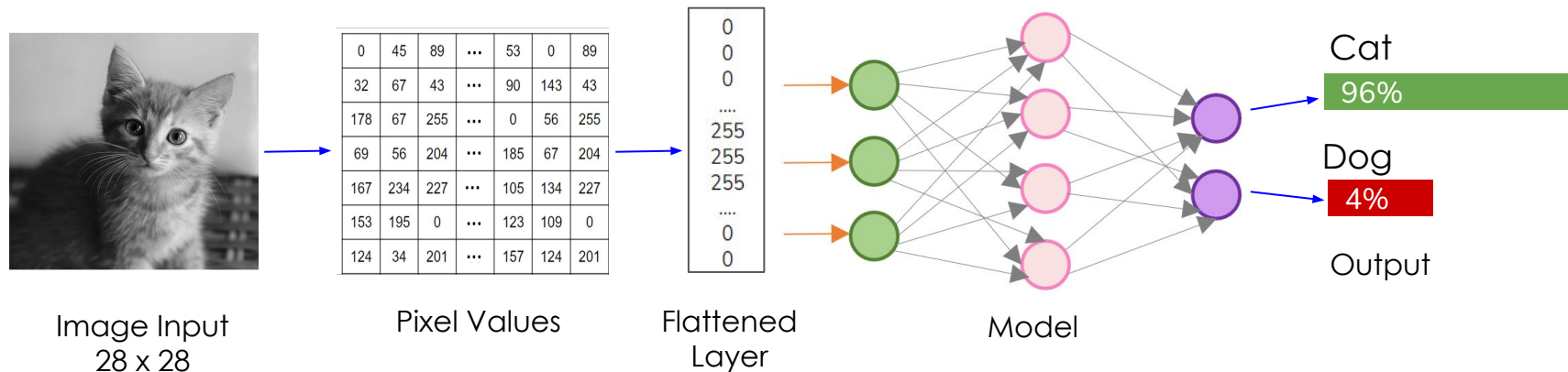


Image classification using ANNs

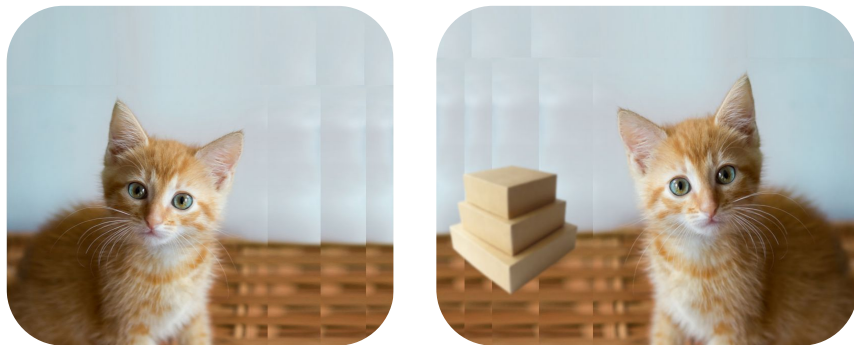
- For that reason, **we will use the Grayscale color space**, which isolates the color information into a single channel.
- In ANNs, one neuron would be used for each input. Here the inputs are the pixel values in an image. For an image with 28×28 pixels, the flattened input layer will have $28 \times 28 = 784$ inputs.
- Every node/element in the output vector refers to an output class. The input sample is labeled according to the class with the highest score.





Challenges with image classification using ANNs

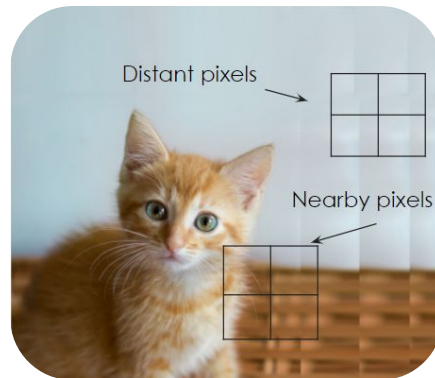
1. Spatial and translational invariance



Translational invariance

Another problem with ANNs is that **they lose spatial information** after getting the image matrix converted into a flattened array. This means ANNs do not leverage the fact that nearby pixels are more strongly related than distant ones.

One of the main disadvantages of ANNs is that **they are not translation invariant**. For example: Even though the cat may be present in the center, the left or the right corner of an image, ANNs would try to learn the location the cat is present as an indicator of its presence, and this may give inconsistent results.

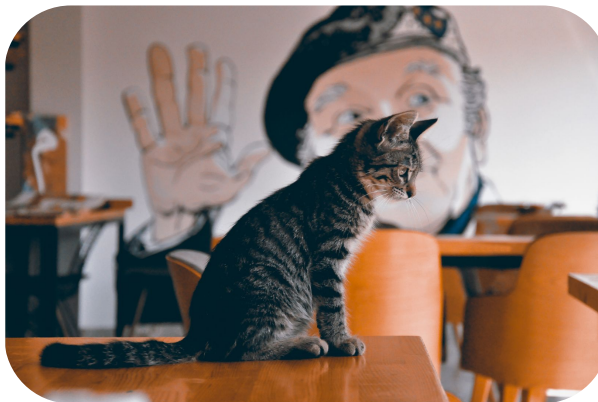


Spatial invariance

2. Extracting important features

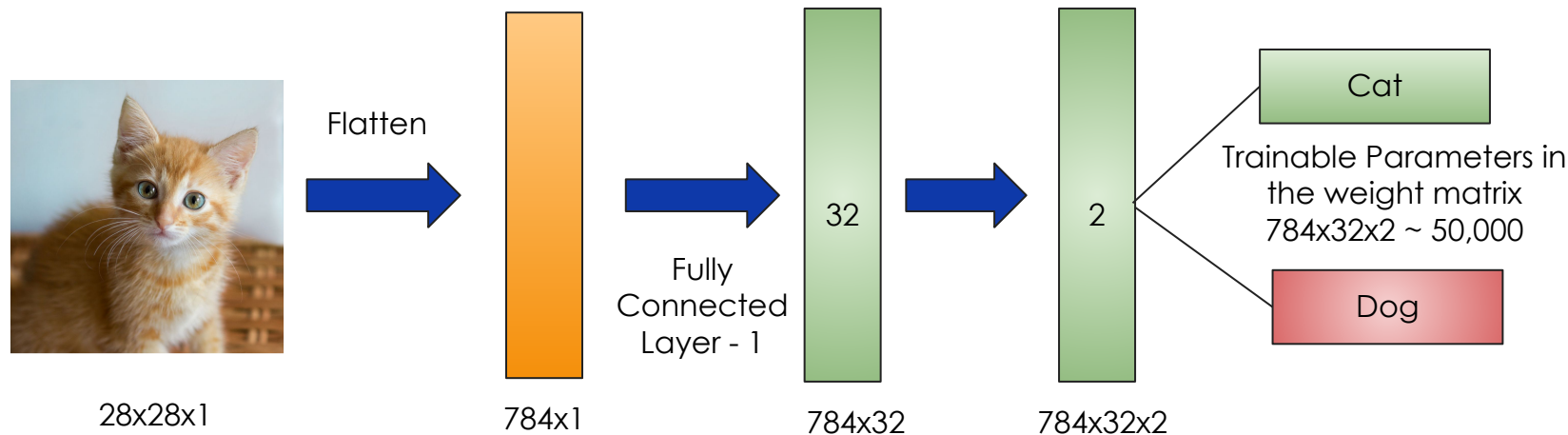
- Yet another disadvantage of ANNs for image classification, has to do with **detecting which features of an image are important and which ones are not**. When detecting the presence of the object, there is no use of looking at the background or other features of an image. There is unfortunately no scope to use this idea in a traditional ANN, which may give importance to every pixel in an image, and will hence try to learn the background of the object as well.

For example: A cat remains a cat whether it is in the garden or on a table. However, an ANN will wrongly learn about the background and assume that a cat appears in a specific background.



3. The computationally expensive nature of ANNs

- To add to that, **ANNs are also very computationally expensive.**
- To use an ANN, we would first have to flatten the input. Using just two fully-connected layers (of 32 and 2 nodes each) on the input image of $28 \times 28 \times 1$, we observe that there are already **about 50,000 trainable parameters**, and this would just increase with an increase in the number of layers.





Why CNN's over ANNs?

Spatial and translational invariance of CNNs

- CNNs have the inbuilt property of handling positional shifts, or translations of a given image. They are **spatially and translationally invariant**.
- In the below example, we have three images of a cat. It obviously remains a cat regardless of whether it appears in the left, the center or the top right corner of the image. Unlike ANNs, **CNNs are able to understand this**, as you would expect for any practical computer vision model.
- So how are CNNs able to accomplish this?



Spatial and translational invariance of CNNs

- CNNs achieve this through the use of **convolutional and pooling layers**.
- Convolutional layers extract the right set of features from the image by applying filters that match those patterns in the image. Since the same filter is applied through a sliding mechanism throughout the whole image, **the features of the image get detected irrespective of their position**.
- The result of the convolutional layer is sent to the pooling layer. The pooling layer reduces the image complexity and size, and extracts the important features from the pooling patch. This achieves the dual task of **eliminating unwanted background features**, as well as the translational invariance of **removing the information about the exact position** of the object in the image.
- This is why CNNs will likely work well even for shifted objects in images, even if the CNN is not explicitly trained on images with such variability.
- Each layer in the CNN introduces some amount of translational invariance, and the more layers we add, the more effective it usually becomes.

Extracting important Features in CNNs

- **CNNs automatically detect or learn the important features of an image without any human supervision.** For example: In an image dataset of handwritten digits, CNNs learn the distinctive features for each digit by themselves, by learning the best filter values through backpropagation.
- Convolutional filters change the input in such a way that only **the important features are extracted.**

In the below images, detecting the presence of the dog does not require extracting information about the background (depicted in red lines). We only need to extract the relevant features of the dog, and ignore all other information. This is achieved using convolution and pooling.



Computational advantage of CNNs

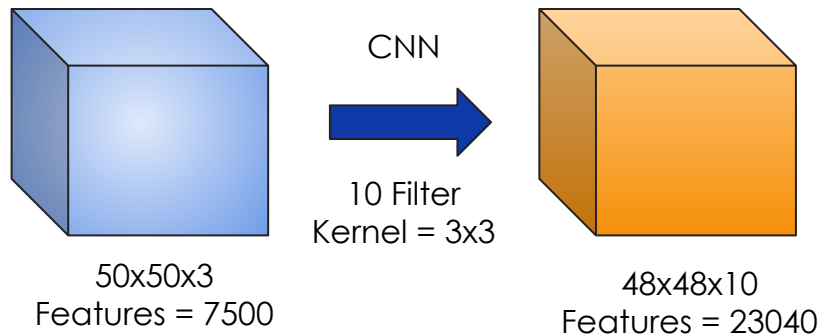
Weight sharing: In the convolution process, we apply the same filter to every patch of the image.

As a result:

- **It reduces the number of weights that must be learned**, which reduces training time and cost.
- **It makes the filter search insensitive to the location** of the important features in the image.

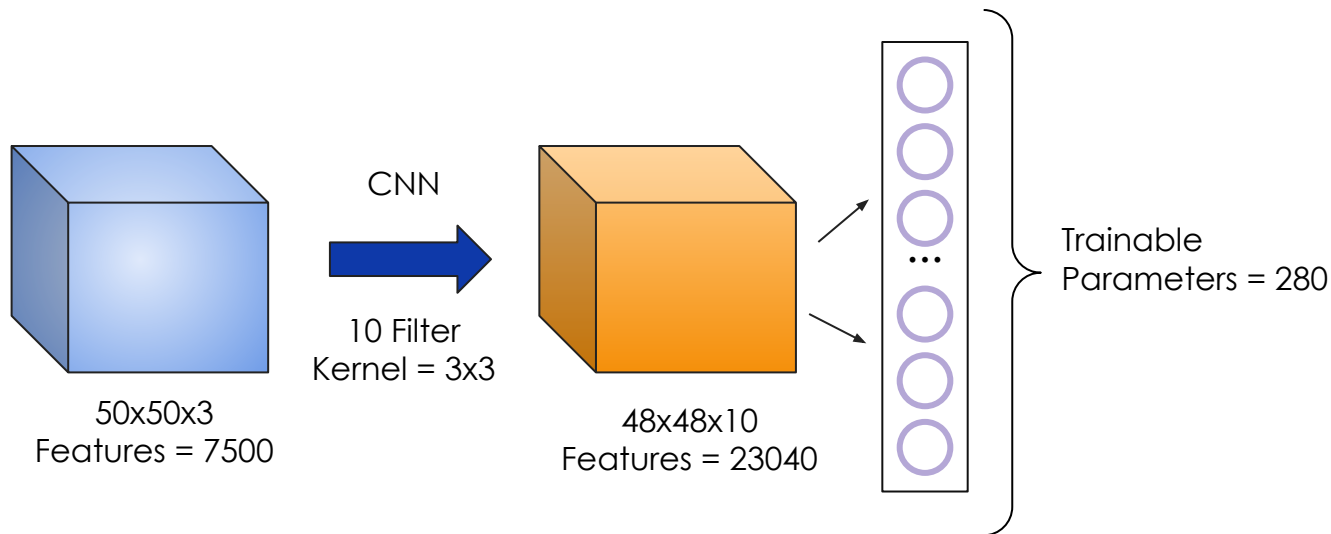
Let's say we have an image of size $50 \times 50 \times 3$ (i.e. 7500 Pixels or Features)

- Taking a CNN Layer with 10 filters and a kernel size of 3×3 , we'll have the given output of size $48 \times 48 \times 10$ size having 23040 features.



Computational Advantage of CNNs

- However unlike in ANNs, the number of trainable parameters still remains a small fraction of the features and does not scale with that large number.
- **Number of Trainable Parameters**
= (filter size x No. of channels + bias) x No. of filters
= $(3 \times 3 \times 3 + 1) \times 10 = \mathbf{280}$



Summary

- CNNs perform better than ANNs in the crucial task of **capturing the relevant features from an image**, by ignoring any spatial and translational transformations.
- The use of filters in a CNN helps us reduce the dimensionality of the image and extract only the important and required information.
- Convolutional filters require exponentially less trainable parameters in comparison to the fully-connected dense layers required in ANNs, and **this gives CNNs a computational advantage over ANNs as well.**



Thank You



The CNN Architecture

Week 1: Computer Vision



Agenda

- Architecture and building blocks
- Building a CNN model

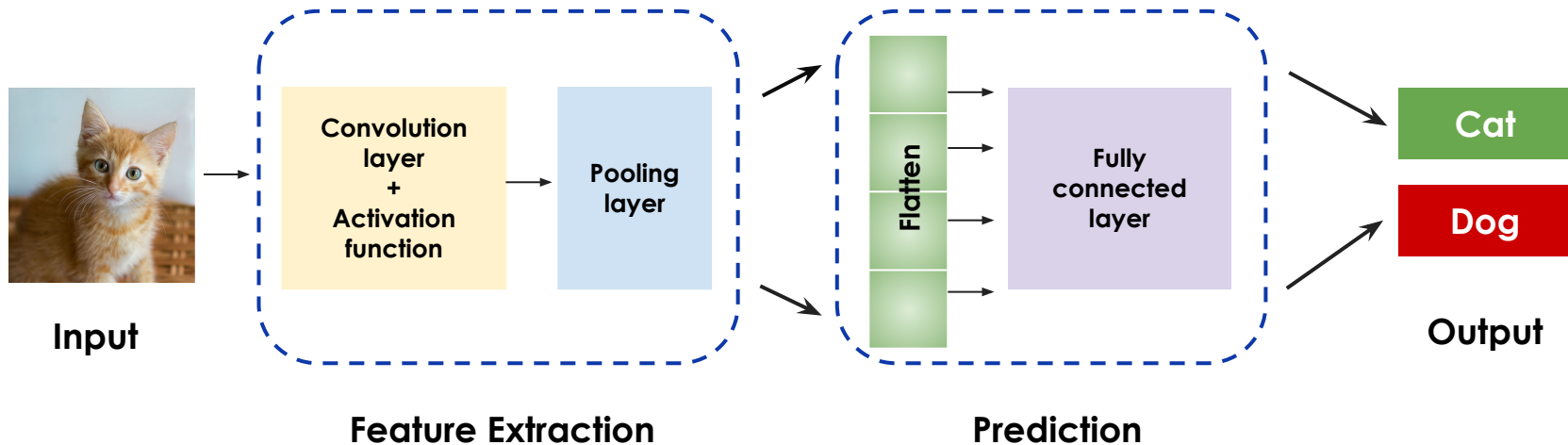


Architecture and building blocks

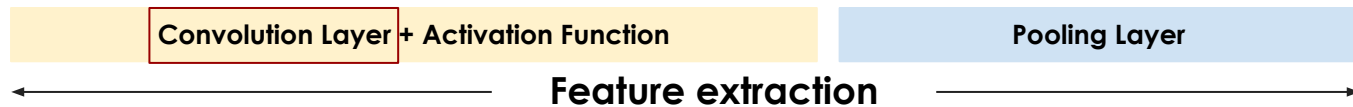
The CNN architecture - Building blocks

The CNN architecture for image classification is comprised of two major parts:

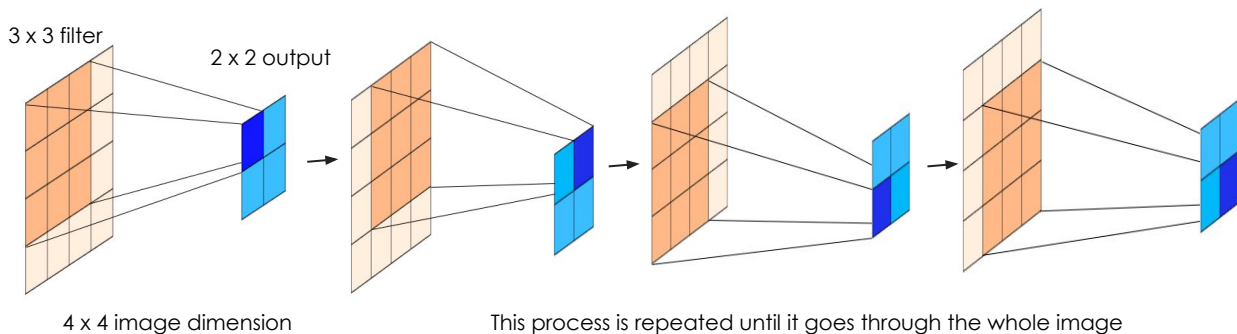
1. **The feature extraction stage** w/ **(a)** Convolution layer + Activation function & **(b)** Pooling layer
2. **The prediction stage** w/ **(a)** Flatten layer & **(b)** Fully connected layer



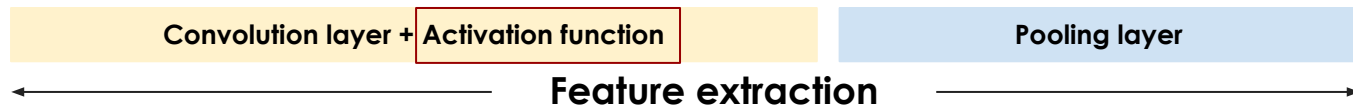
The convolutional layer



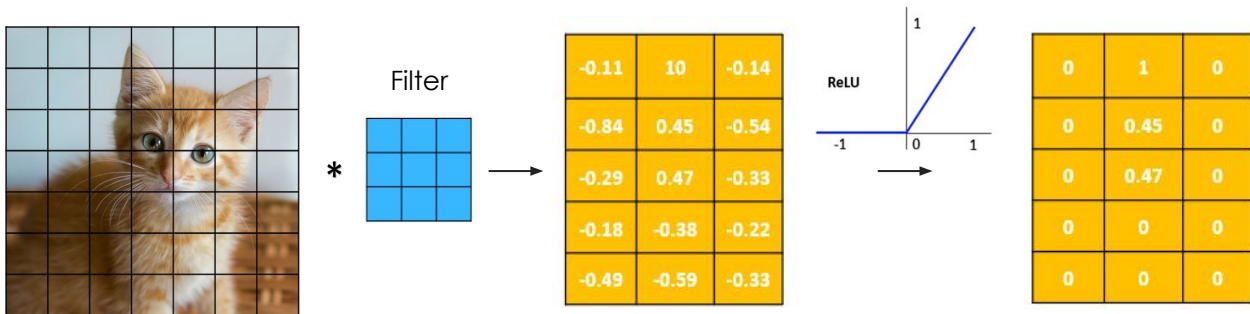
- In the convolutional layer, a filter is applied on an input image to build a feature map.
- Filters can be custom made. In convolutional neural networks though, the values of the filters are learned through training to determine the important features that can categorize the image more accurately. We simply pass the number of filters and their sizes to the CNN, and the model learns the filter values to develop various feature maps that capture the presence of the features detected.



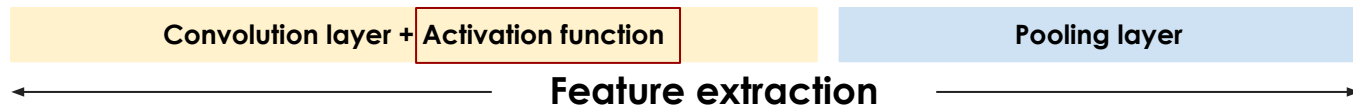
The activation function



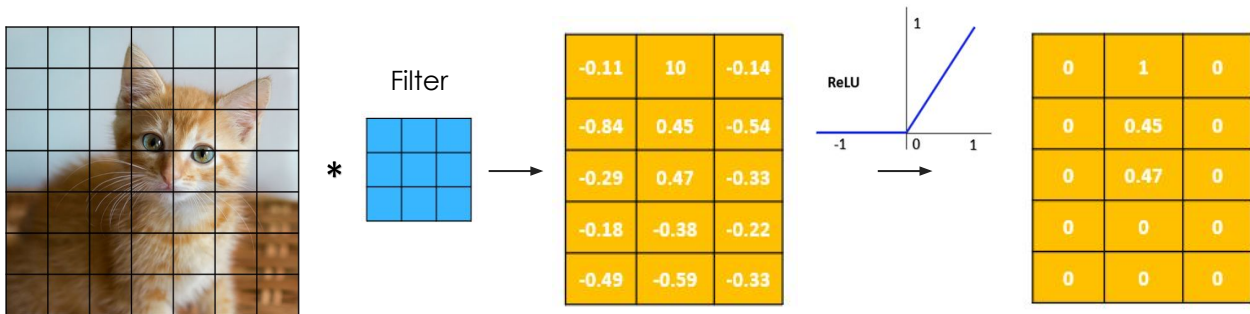
- The next step is the Activation Function. In CNNs, the **ReLU activation function** is used to incorporate a non-linear component into neural networks.
- The ReLU function is used in CNNs because the convolution operation is essentially a linear operation (a sum of element-wise products) between the image and the filter, and ReLU adds the non-linearity required for this operation to be able to detect complex non-linear boundaries in the image.



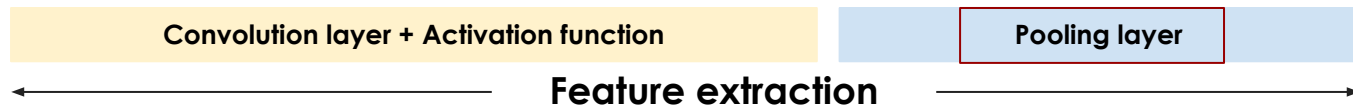
The activation function



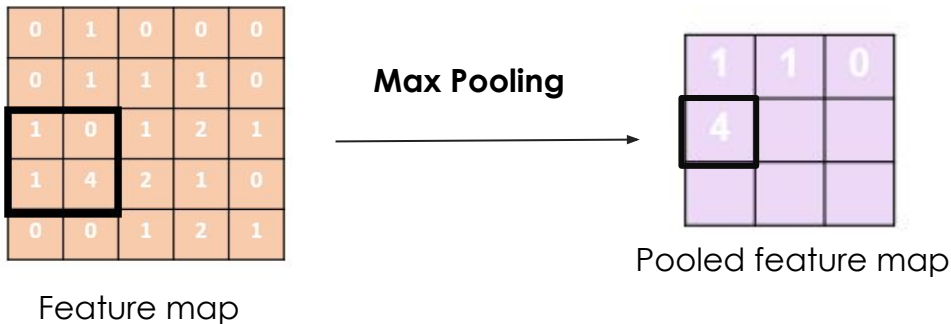
- One advantage of using **ReLU (Rectified Linear Unit)** is that if there are any negative pixels, the function will convert it into zero. This is useful for visualizing these image maps, since negative pixels have no meaning, and it makes subsequent computations more efficient since zeros are easy to work with.



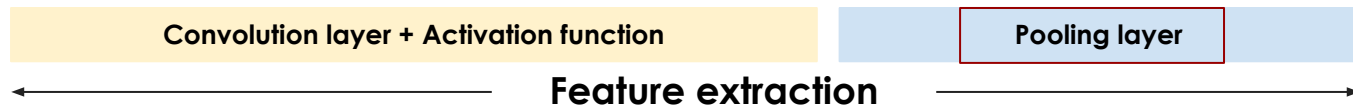
The pooling layer



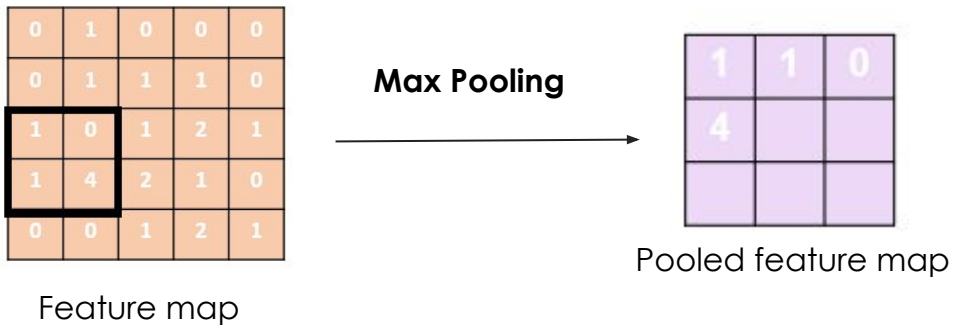
- The next step in the Feature Extraction stage is **Pooling**. The pooling layer helps in removing unwanted features from the image. In doing so, it reduces the size of the image and hence decreases the computational cost of the model.
- CNNs can use '**Max Pooling**' to create a pooled feature map using only the maximum values of each patch, and dispose of the unnecessary pixel information.
- An important question here is - **Do we lose information in this process?**
The answer is **Yes**. The final feature map includes fewer cells and thus less information than the original input image. But, the purpose of this step is to **discard irrelevant features** so that the network can do its job more efficiently.



The pooling layer



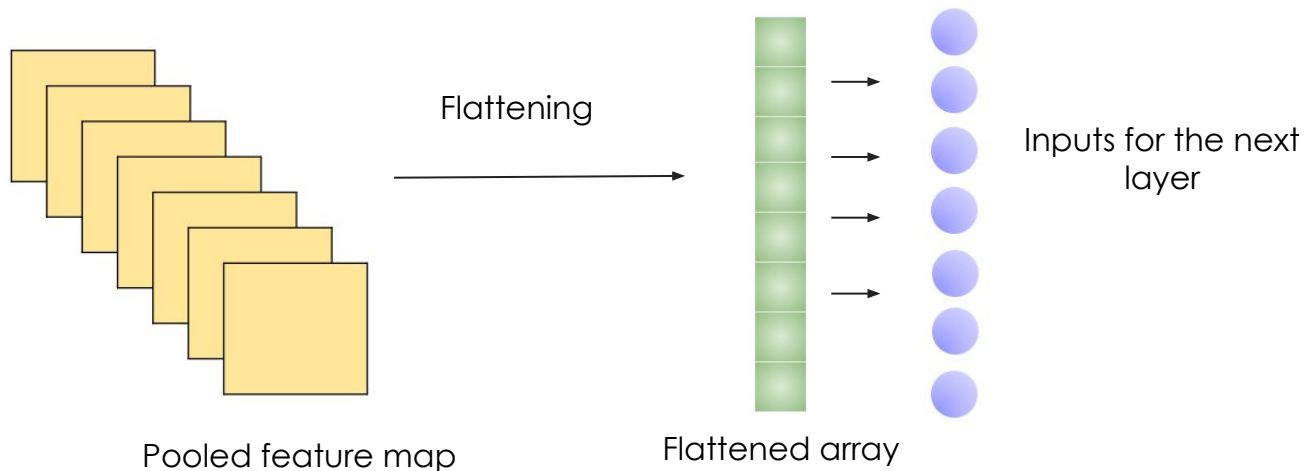
- This process is what's responsible for the "spatial invariance" property of CNNs. Pooling also reduces the size of the images, and hence the number of parameters, which minimises the likelihood of "overfitting", which neural networks are often susceptible to.



The Flatten Layer



- After the Feature Extraction stage, we move to the Prediction stage of the CNN. The first step of the Prediction stage is **the Flatten layer**.
- In this layer, the CNN literally **flattens** the pooled feature map into a column vector, and the output is used in a standard artificial neural network configuration, which are the fully connected layers of the CNN.



The Fully connected layer



- So far, we have learned about the convolution operation, the activation function, pooling and flattening. The final stage of the process is the fully connected layer, where the information from the previous layers is sent to an artificial neural network architecture with Dense (fully connected) layers.
- The aim of this step is to take the input and combine the features into a wider variety of attributes that make the convolutional network more capable of categorising images, which is the sole purpose of building a convolutional neural network.





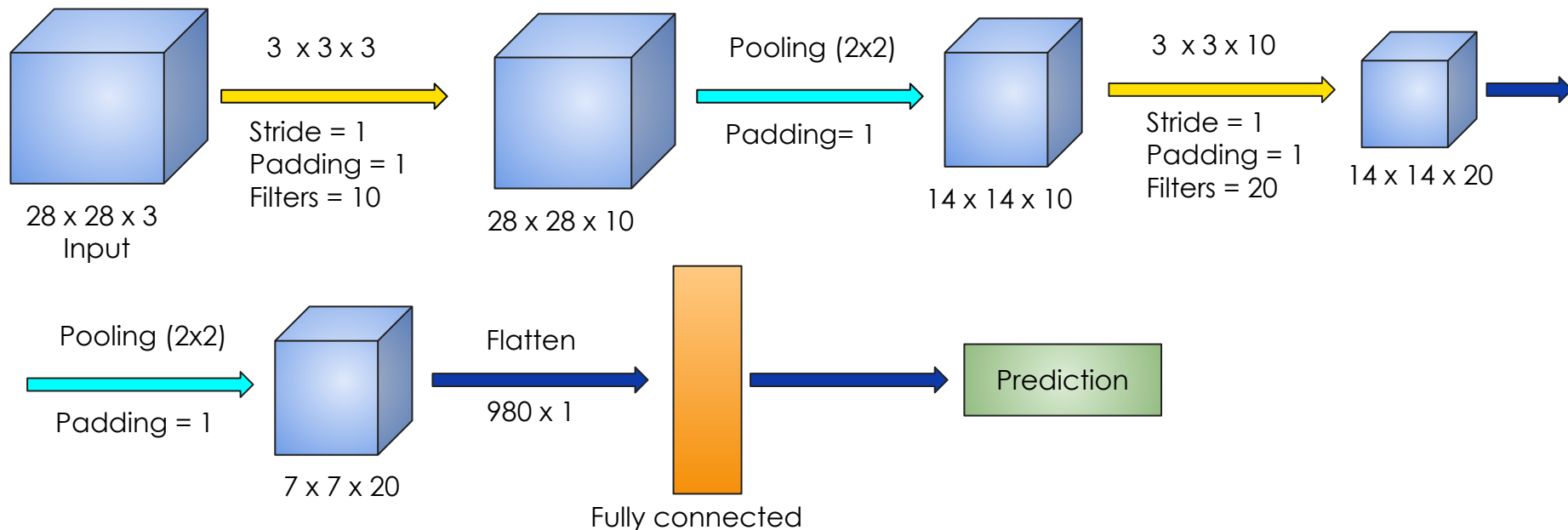
Building a CNN Model

Let's build a Convolutional Neural Network

Let's say we want to build a simple Convolutional Neural Network (CNN) for a prediction problem.

For example: Let's say we need to classify an image as a dog or a cat.

We can build a simple CNN with convolution and pooling layers to solve this problem.



Summary

To summarize, let's go over what we've learnt about convolutional neural networks in a nutshell:

- We are starting off by applying filters to input images in the form of a convolutional layer.
- We break up the linearity of the convolution operation, using the ReLU activation function.
- We reduce the image dimension using pooling, for isolating the important features and computational efficiency.
- We flatten the feature map and feed it into a fully-connected neural network to generate the final predictions.

Throughout this process, the trainable parameters of the CNN - the weights and filter values, are trained and continuously updated through backpropagation so that the CNN can achieve its best possible performance in image prediction tasks.



Thank You