**Identify medication based on the imprint code, colour, shape and Form.**

**Imprint Code**

Each type of medication will also be imprinted with a unique code. These codes can include a combination of numbers and letters or the name of the drug. In some cases, you might also see a logo. Some of the letters and numbers can be hard to distinguish, especially on very small pills.

**Shape**

Medication comes in all sorts of shapes. You might be most familiar with round or oblong pills and capsules. However, some medication is unusually shaped in the form of squares, rectangles, diamonds, triangles, pentagons, hexagons, heptagons, and octagons.

**Colour**

Each type of pill has a standard colour. Some are very familiar like brown Advil tablets or blue Viagra pills. Capsules, pills, tablets, and caplets do not need to be one solid colour, though. Pills might be one colour on one side and a different colour on the other side, or capsules might be made up of two different coloured pieces. Pills and caplets might also have a coloured pattern such as specks of red on a solid white background. Different forms of a medication might come in different colours. For instance, it might be white in pill form and green in capsule form.

**Form**

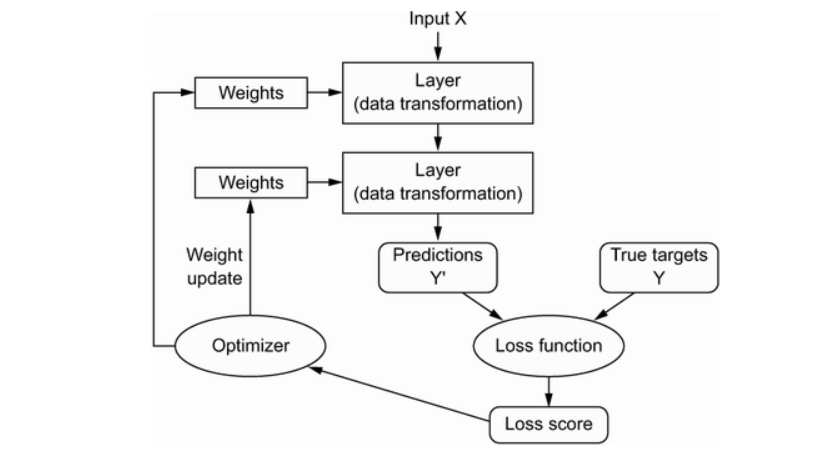
The form refers to whether the medication is a tablet, capsule, or other type of oral medication.

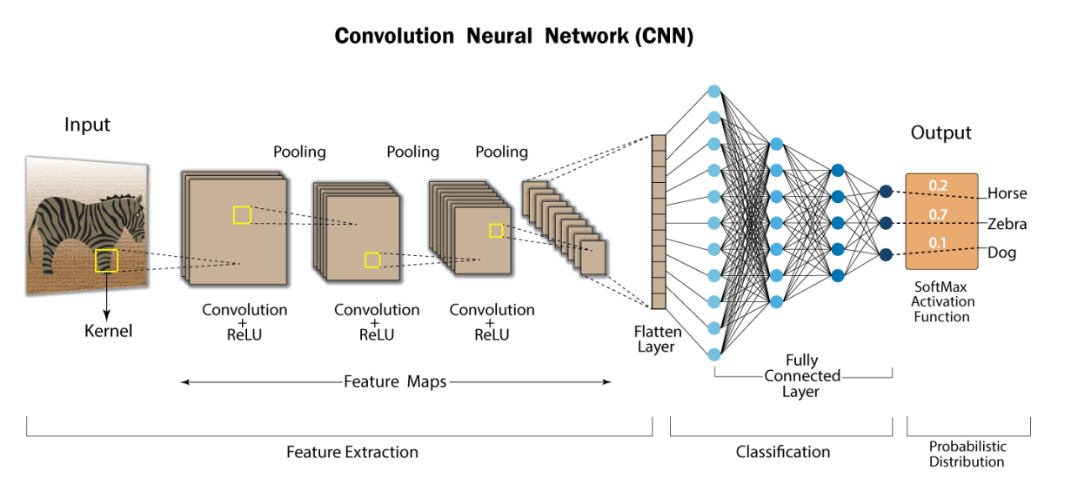
**Scoring**

Pills may have scores, which are like light lines cut into them. There may be one, multiple, or none, depending on the medication. They can appear with the imprint code or on the reverse side.

**Image classification:**

Image classification involves assigning labels or classes to input images. It is a supervised learning task where a model is trained on labelled image data to predict the class of unseen images. CNN are commonly used for image classification as they can learn hierarchical features like edges, textures, and shapes, enabling accurate object recognition in images. CNNs excel in this task because they can automatically extract meaningful spatial features from images. Here are different layers involved in the process:





**Model Layers:**

**Rescaling**: A preprocessing layer which rescales input values to a new range. This layer rescales every value of an input (often an image) by multiplying by scale and adding offset.

Input shape:Arbitrary.

Output shape:Same as input.

Arguments

scale: Float, the scale to apply to the inputs.

offset: Float, the offset to apply to the inputs.

**Conv2D (Convolutional Layer)** :

Convolution is a mathematical way of combining two signals to form a third signal, to roll together or circulate involvedly.

It tries to learn the feature representation of the inputs, whether it be the images of cats vs dogs or digits. For computing the different feature maps, it is composed of several kernels/matrix are used. So, a filter/kernel of (n\*n) matrix depends on the type of problem we are solving, and then it is applied to the input data (or image) to get the convolutional feature. This convolution feature is then passed on to the next layer after adding bias and applying any suitable activation function.

(It tries to learn the feature representation of the inputs) This layer creates a convolution kernel (convolved with the layer input to produce a tensor of outputs).  If use\_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

**MaxPooling2D:** Max pooling operation for 2D spatial data. The main purpose of pooling is to reduce the size of feature maps, which in turn makes computation faster because the number of training parameters is reduced. The pooling operation summarizes the features present in a region, the size of which is determined by the pooling filter. The pooling layer is placed between the convolutional layers. It is used for achieving shift invariance which is achieved by decreasing the resolution of the feature maps. The widely used pooling operations are average pooling and max pooling. Basically, reducing the number of connections between convolutional layers, lowers the computational burden on the processing units.

**Flatten**: The output of the last pooling layer is flattened and connected to one or more fully connected layers. Flattens the input. Does not affect the batch size and Changes shape of image structure.These layers function as traditional neural network layers and classify the extracted features. The fully connected layers learn complex relationships between features and output class probabilities or predictions.

**Dense(Output Layer)**

The output layer represents the final layer of the CNN. It consists of neurons equal to the number of distinct classes in the classification task. The output layer provides each class’s classification probabilities or predictions, indicating the likelihood of the input image belonging to a particular class.

The last layer of CNNs is an output layer that makes final predictions. For classification tasks, the **Softmax** function is commonly used where multiple classes are targeted for prediction (Ex. MNIST Dataset) and the **Sigmoid** function is used for binary classification (Ex-Cats vs Dogs).

implements the operation: output = activation (dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use\_bias is True). These are all attributes of Dense.

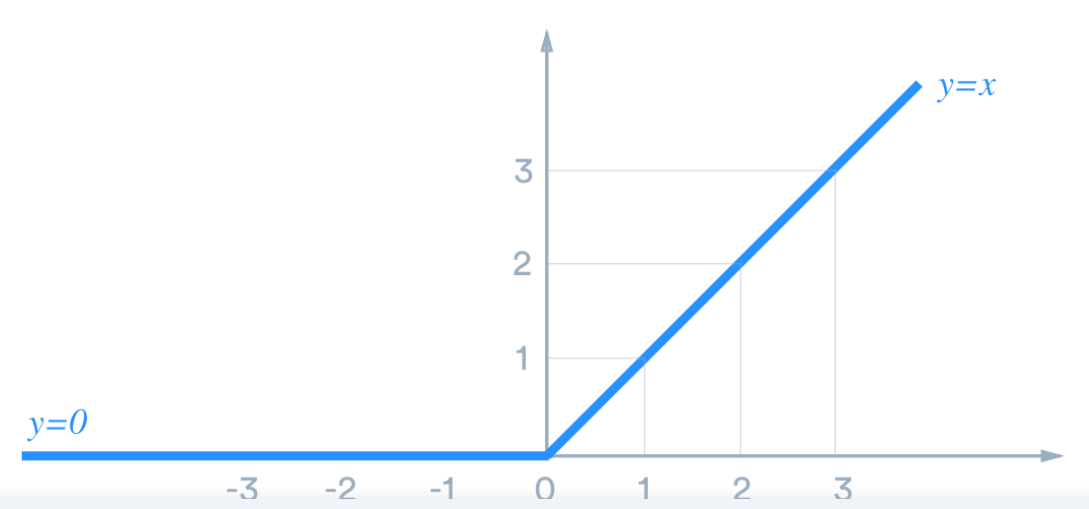
**Note**: If the input to the layer has a rank greater than 2, then Dense computes the dot product between the inputs and the kernel along the last axis of the inputs and axis 0 of the kernel (using tf.tensordot). For example, if input has dimensions (batch\_size, d0, d1), then we create a kernel with shape (d1, units), and the kernel operates along axis 2 of the input, on every sub-tensor of shape (1, 1, d1) (there are batch\_size \* d0 such sub-tensors). The output in this case will have shape (batch\_size, d0, units).

Besides, layer attributes cannot be modified after the layer has been called once (except the trainable attribute). When a popular kwarg input\_shape is passed, then keras will create an input layer to insert before the current layer. This can be treated equivalent to explicitly defining an InputLayer.

Dense

**Activation Function**

Any selected activation function hugely impacts the performance of a convolutional neural network for a given problem. For CNN’s hidden layers, Re Lu is the preferred activation function because of its simple differentiability and fastness compared to other activation functions like tanh and sigmoid. Re Lu is typically followed after convolution operation. Other names in a list of activation functions include Sigmoid, softmax, Leaky Re Lu, ELU, etc.

****

**optimizer** = 'adam',

**loss** = tf.keras.losses.SparseCategoricalCrossentropy

In particular, deep learning has enabled the following breakthroughs, all in historically difficult areas of machine learning:

• Near-human-level image classification

• Near-human-level speech transcription

• Near-human-level handwriting transcription

• Dramatically improved machine translation

• Dramatically improved text-to-speech conversion

• Digital assistants such as Google Assistant and Amazon Alexa

• Near-human-level autonomous driving

• Improved ad targeting, as used by Google, Baidu, or Bing

• Improved search results on the web

• Ability to answer natural language questions

• Superhuman Go playing