Tensorflow CNN layers

All of these examples of layers can be found within the library tf.keras.layers

1. **Dense Layer (Fully connected layer)** - most common layer type used in feed-forward neural networks. It connects every layer in the input to every layer in the output and applies a linear transformation function followed by a non-linear activation function.

**# Create a Dense layer with 64 neurons and ReLU activation**

**dense\_layer = Dense(units=64, activation='relu')**

* Units - how many neurons you want in your connected network
* Activation - the type of activation function you want to layer to use.

1. **Convolutional layer** - commonly used in computer vision tasks, can learn spatial features from input images. Performs a convolution operation on an input image with a set of learnable filters.

**# Create a Conv2D layer with 32 filters, each with a size of 3x3 and ReLU activation**

**conv\_layer = Conv2D(filters=32, kernel\_size=(3,3), activation='relu')**

* Filters - The depth of the output
* Kernel\_size - The size of the kernel/mask used for the convolution
* Activation - The type of activation function you want the layer to use
* Padding - The type of padding that will be added to the image

1. **Pooling layer** - reduces the spatial dimensions of the input by summarising groups of values. Common types of pooling layers include Max pooling (taking the maximum of each area) and average pooling (taking an average).

**# Create a MaxPooling2D layer with a pool size of 2x2**

**pooling\_layer = MaxPooling2D(pool\_size=(2,2))**

* Pool\_size - The size of the pooling window
* Strides - The stride length for the pooling window
* Padding - The type of padding to use

1. **Recurrent Layer -** Layer used in sequence modelling tasks such as NLP and time series analysis.

**# Create an LSTM layer with 64 units**

**lstm\_layer = LSTM(units=64)**

* Units - The dimensionality of the output space
* Activation - the activation function to use

1. **Embedding Layer:** Used in Natural Language Processing tasks to map categorical inputs such as words to continuous vectors. It learns a set of embeddings which are dense vectors that capture semantic meaning, and applies them to the input.

**model.add(keras.layers.Embedding(10000, 16))**

6)

7)

8)

9)

10)

11)

12)

13)

14)

15)

16)