**MODEL SUMMARY**

As the task is to classify the image, the model chosen is Convolutional Neural network. This model is mainly used to identify satellite images, process medical images, forecast time series and detect anomalies.

CNN has multiple layers that extract features from data.

1) Convolution Layers

a. Core building block

b. Requires few components like input data, filter and a feature map

c. Feature Detector also known as kernel or filter moves across the receptive field of images to check if the feature is present. This is called convolution. Filter is applied to an area of the image and dot product is calculated and afterwards the filter shifts by a stride and repeat the process until the kernel has moved all over the entire image

2) Activation Functions

3) Pooling Layer

Also referred to as down sampling layers, they aim to reduce the number of parameters in the input data.

Two primary types of pooling layers:

* Max pooling: Selects the maximum value within a pixel set and sends it to the output array.
* Average pooling: Computes the average value within the receptive field and forward it to the output array.

CNNs, as feedforward neural networks, excel in analysing visual images by processing data in a grid-like structure. The convolutional layer contains various filters that execute convolution operations. Each image is treated as a matrix of pixel values, and the filter matrix slides over the image matrix, computing dot products to generate a convolution feature matrix.

**TRAINING PROCESS**

* **Image Processing**
* Utilize OpenCV to load each image.
* Convert images to the RGB color space.
* Resize images to a uniform size of (128, 128).
* Convert the images into NumPy arrays**.**
* **Model Architecture**

Employ a sequential CNN model comprising:

* A convolutional layer with 32 filters, a 3x3 kernel size, and ReLU activation.
* A max-pooling layer with a 2x2 pool size.
* A flatten layer to transform the 2D feature map into a 1D vector.
* A dense layer featuring 256 units and ReLU activation.
* A dropout layer with a 0.1 dropout rate to prevent overfitting.
* Another dense layer having 512 units and ReLU activation.
* An output layer with 5 units using softmax activation for multi-class classification.
* **Model Compilation**

Compile the model using Adam optimizer and sparse categorical cross-entropy loss, with accuracy as the metric**.**

* **Model Training:**
* Split the dataset for training, allocating 70% for training and 30% for testing using the train\_test\_split method from scikit-learn**.**
* Normalize both training and testing data using TensorFlow to scale pixel values between 0 and 1.
* Train the model over approximately 200 epochs with a 0.1 validation split.

**INSIGHTS**

Despite attaining an impressive 84% accuracy over 200 epochs, the model's effectiveness is likely affected by the dataset's small scale, containing only 150 images. This limited dataset makes it difficult for the model to grasp broad patterns, resulting in a tendency to overfit. Enhancing the model's performance would involve expanding and diversifying the training dataset while experimenting with various regularization methods.