General steps to create an Image Classifier AI model:

1. **Collect Data**: Gather a dataset of images, each labeled with their respective class.
2. **Preprocess Data**: Normalize images, handle missing data, and split into training and testing sets.
   1. **Image Reading**: Load the image files.
   2. **Image Resizing**: Resize images to ensure they all have the same dimensions for easier and quicker processing (smaller images also scaled up).
   3. **Normalization**: Scale pixel values to a range of 0 to 1 (smaller values make the process more efficient).
   4. **Data Augmentation**: Apply techniques like rotation, zooming, flipping to increase dataset size and reduce overfitting.
   5. **Grayscale Conversion**: If color is not important, convert images to grayscale to reduce computational complexity.
   6. **Splitting Dataset**: Divide the data into training, validation, and testing sets.
   7. **Label Encoding**: Convert categorical labels into numerical form.
3. **Choose Model Architecture**: Decide on a model structure (e.g., CNN).
4. **Train Model**: Use a suitable optimizer and loss function to train the model on the training set.
5. **Evaluate Model**: Test the model’s performance on the testing set.
6. **Optimize**: Tune hyperparameters or adjust the model architecture based on performance.

How does a CNN work?

1. Feature Extraction: The first step of a CNN is ‘feature extraction’. It involves multiple different layers, and a single type of layer may occur multiple times. The details are:
   1. Convolutional layers:
      * Each convolutional layer involves a number of ‘filters’, which are applied onto the image (called convolution) like a sliding window.
      * The resultant pixels form a ‘feature map’ (size may vary from the input image).
      * Padding during convolution.
      * An activation function may be applied onto the feature map to introduce non-linearity.
      * The filters have variable weights for each pixel (and for each input channel of that pixel e.g., RGB).
      * Each filter also has a bias.
      * The bias and weights are updated during the learning process.
      * Each convolutional layer may produce multiple feature maps (but the size of each feature map is the same).
      * We can say that after a convolutional layer, we could have a 3D output-volume/image (multiple feature maps, each having 2Ds).
   2. Pooling layers:
      * Applied after convolution layers usually.
      * Reduces the overall dimensions of the input-volume (which we said could be 3D) to make processing faster. It also helps in preventing overfitting and also helps in making a similar output feature map even if the target object shifts around in the input images.
      * Also similar to sliding a window around the feature map and producing a single value.
      * GlobalMaxPooling, AveragePooling, and MaxPooling.

Example of Convolution and Pooling Layers below.

Note: After any of these layers, we can ‘drop’ some of the output values, i.e., make a certain percentage of output pixels of the feature maps 0 (note that each output value was actually the output of a neuron).

Moreover, a feature extraction part of the layer may have multiple convolution layers followed by pooling layers, which may then once again be followed by convolution layers and pooling layers etc.

* 1. Flattening:
     + Finally, the output produced by the previous layer is flattened (multiple ways to do so) and fed to the classification part of the CNN.

1. Classification: An Artificial Neural Network (ANN) that classifies the input instance (flattened output) into a specific class.

Example:

1. **Input**: The input is an image of size 32x32x3 (width x height x color channels).
2. **First Convolutional Layer**: This layer might have 64 filters of size 3x3. After applying these filters, we get 64 feature maps of size 30x30 (assuming stride of 1 and no padding). So, the output volume of this layer is 30x30x64.
3. **First Activation Function**: A non-linear activation function, such as ReLU, is applied to the output volume of the first convolutional layer. This introduces non-linearity into the model, allowing it to learn more complex patterns.
4. **First Pooling Layer**: A 2x2 max pooling layer reduces the spatial dimensions by half, resulting in an output volume of 15x15x64.
5. **Second Convolutional Layer**: This layer might have 128 filters of size 3x3. After applying these filters, we get 128 feature maps of size 13x13 (again assuming stride of 1 and no padding). So, the output volume of this layer is 13x13x128.
6. **Second Activation Function**: Again, a non-linear activation function is applied to the output volume of the second convolutional layer.
7. **Second Pooling Layer**: A 2x2 max pooling layer again reduces the spatial dimensions by half, resulting in an output volume of 6x6x128.