EMERGING METHODS FOR EARLY DETECTION OF FORESTFIRES

MODEL BUILDING

ADDING DENSE LAYERS

Date	09 November 2022
Team ID	31445-1660200428
Project Name	Emerging Methods for Early Detection of Forest Fires

Adding Dense Layers:

- Layers in the deep learning model can be considered as the architecture of the model.
- There can be various types of layers that can be used in the models.
- All of these different layers have their own importance based on their features.
- Like we use LSTM layers mostly in the time series analysis or in the NLP problems, convolutional layers in image processing, etc.
- A dense layer also referred to as a fully connected layer is a layer that is used in the final stages of the neural network.
- This layer helps in changing the dimensionality of the output from the preceding layer so that the model can easily define the relationship between the values of the data in which the model is working.

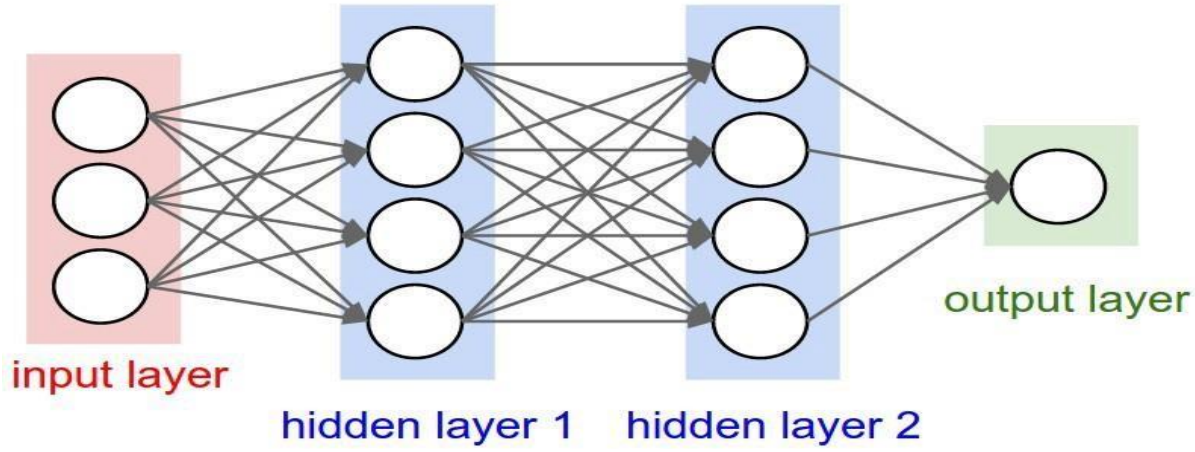
- A dense layer is a layer that is deeply connected with its preceding layer which means the neurons of the layer are connected to every neuron of its preceding layer. This layer is the most commonly used layer in artificial neural network networks.
- Dense Layer is used to classify image based on output from convolutional layers. Each Layer in the Neural Network contains neurons, which compute the weighted average of its input and this weighted average is passed through a non-linear function, called as an “activation function”.

The name suggests that layers are fully connected (dense) by the neurons in a network layer. Each neuron in a layer receives input from all the neurons present in the previous layer. Dense is used to add the layers.

Task 1: Adding Hidden layers:

In artificial neural networks, hidden layers are required if and only if the data must be separated non-linearly.

A hidden layer in an artificial neural network is a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function.



They allow you to model complex data thanks to their nodes/neurons. They are “hidden” because the true values of their nodes are unknown in the training dataset. In fact, we only know the input and output. Each neural network has at least one hidden layer. The first layer is the input layer and the last one is the output layer. Whatever comes in between these two are the hidden layers.

Adding a hidden layer between the input and output layers turns the Perceptron into a universal approximator, which essentially means that it is capable of capturing and reproducing extremely complex input–output relationships.

This step is to add a dense layer (hidden layer). We flatten the feature map and convert it into a vector or single dimensional array in the Flatten layer. This vector array is fed it as an input to the neural network and applies an activation function, such as sigmoid or other, and returns the output.

- Hidden layers — intermediate layer between input and output layer and place where all the computation is done.

```
In [ ]: #add hidden layer
        model.add(Dense(output_dim=150,init='uniform',activation='relu'))
```

The hidden layers of a CNN typically consist of **convolutional layers, pooling layers, fully connected layers, and normalization layers**. Here it simply means that instead of using the normal activation functions defined above, convolution and pooling functions are used as activation functions.

Key terms:

- In it is the weight initialization; initialization function is the network initialization function which sets all the weights and biases of a network to values suitable as a starting point for training.

- Units, which denotes is the number of neurons in the hidden layer.
- Activation function defines the output of input or set of inputs or in other terms defines node of the output of node that is given in inputs. They basically decide to deactivate neurons or activate them to get the desired output. It also performs a nonlinear transformation on the input to get better results on a complex neural network. Its nodes here just pass on the information (features) to the hidden layer.

You can add many hidden layers.

Task 2: Adding output layer:

The output layer is the final layer in the neural network where desired predictions are obtained. There is one output layer in a neural network that produces the desired final prediction. It has its own set of weights and biases that are applied before the final output is derived.

The output layer is the final layer in the neural network where desired predictions are obtained. There is one output layer in a neural network that produces the desired final prediction. It has its own set of weights and biases that are applied before the final output is derived.

This step is to add a dense layer (output layer) where you will be specifying the number of classes your dependent variable has, activation function and weight initializer as the arguments. We use add () method to add dense layers. In this layer, no need of mentioning input dimensions as we have mentions them in the above layer itself.

- Output layer — produce the result for given inputs.

```
In [ ]: #add output layer
        model.add(Dense(output_dim=1,activation='sigmoid',init='uniform'))
```

- if you have only one or two class in output put “output_dim = 1” and “activation = sigmoid”. If you have more number of classes for supposing there are 3 classes then put “output_dim = 3” and “activation = softmax”.