**Predicting Effective Diffusivity using CNN**

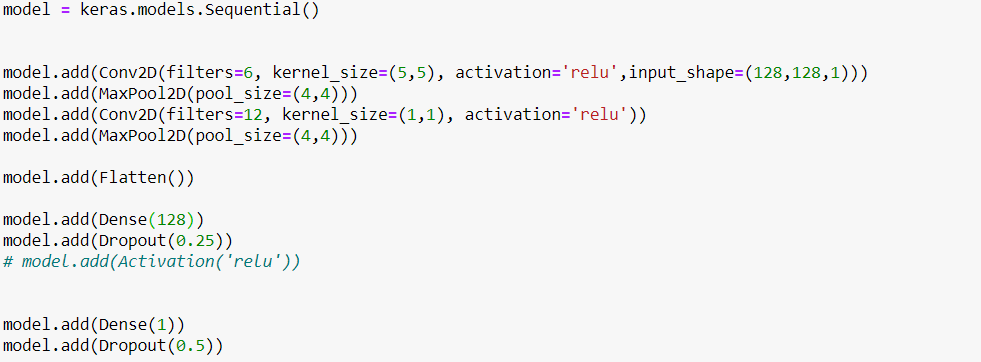
**Workflow**

1. **Data Loading**
   1. Binary Images are read from excel file using OpenCV library in Python (x)
   2. Diffusivity is read from excel file using pandas library in Python (y)
2. **Normalization and Train/Test split**
   1. Since diffusivity values are very small, we normalize it by a normalizing factor (1.958\*10^-9)
   2. Both x and y data is split into train (90%) and test (10%) after normalization
3. **Model Development**
   1. A 2 layer convolutional neural network is developed with functionalities like Convolutions, Pooling, Dropout, Activations, etc. Model is discussed in detail in the next section.
4. **Model Training**
   1. Training is performed on 180 images using Adam Optimizer and learning rates of 10^-3 and 10^-4
   2. Best model checkpoint is saved amongst 50 epochs of training
5. **Evaluation on test set**
   1. Model is saved as a h5 file and the weights are loaded to evaluate the model on test set (20 images) and different models are compared to check the performance
   2. Performance metric – Mean Absolute Percentage Error (MAPE)

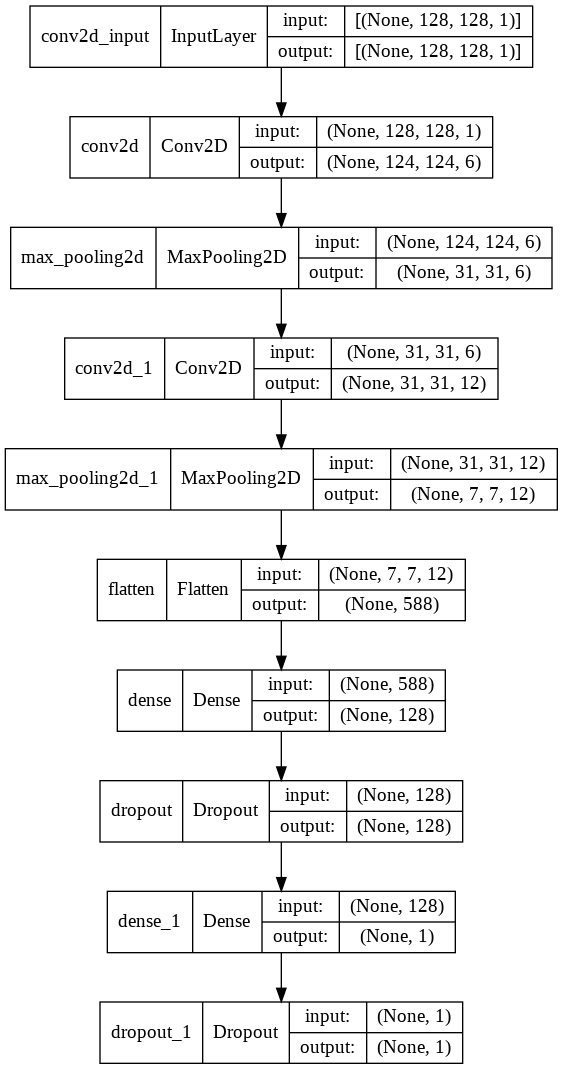
**Model Description – Terminologies**

1. **Sequential model** – this model is used for a plain stack of layers where each layer has exactly one input tensor and one output tensor
2. **Convolution layers** - This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs.
3. **Max Pooling layers** - Downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window for each channel of the input.
4. **ReLU Activation function** – Applies rectified linear unit activation : max(x, 0) on the input tensor.
5. **Padding** - one of "valid" or "same" (case-insensitive). "valid" means no padding. "same" results in padding with zeros evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input.
6. **Filters** - the dimensionality of the output space (i.e. the number of output filters in the convolution).
7. **Kernel size** - An integer specifying the height and width of the 2D convolution window.
8. **Dropout** - The Dropout layer randomly sets input units to 0 during training time, which helps prevent overfitting.
9. **Dense layers** - Dense layer 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.

**Sample model development code**



**Sample Model plot**



**Results**

Old data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Padding** | **ReLU** | **Kernel size** | **Learning Rate** | **Train MAPE** | **Test MAPE** |
|  |  |  |  |  |  |  |
| 1 | Valid | Yes | 5x5 | 10e-3 | 68.50 | 73.52 |
| **2** | **Valid** | **No** | **5x5** | **10e-3** | **53.71** | **58.41** |
| 3 | Same | Yes | 5x5 | 10e-3 | 58.67 | 81.72 |
| 4 | Same | No | 5x5 | 10e-3 | 67.84 | 52.76 |
| 5 | Valid | Yes | 5x5 | 10e-4 | 55.72 | 97.99 |
| 6 | Valid | Yes | 3x3 | 10e-3 | 51.27 | 89.01 |
| 7 | Valid | Yes | 7x7 | 10e-3 | 59.15 | 108.67 |

New Data

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Padding** | **ReLU** | **Kernel size** | **Learning Rate** | **Train MAPE** | **Test MAPE** |
|  |  |  |  |  |  |  |
| 1 | Valid | Yes | 5x5 | 10e-3 | 47.56 | 51.21 |
| **2** | **Valid** | **No** | **5x5** | **10e-3** | **45.31** | **44.61** |
| 3 | Same | Yes | 5x5 | 10e-3 | 44.43 | 46.59 |
| 4 | Same | No | 5x5 | 10e-3 | 48.83 | 51.75 |
| 5 | Valid | Yes | 5x5 | 10e-4 | 45.96 | 54.70 |
| 6 | Valid | Yes | 3x3 | 10e-3 | 43.52 | 53.38 |
| 7 | Valid | Yes | 7x7 | 10e-3 | 55.73 | 59.29 |