Natural Images Recognition using CNN

INFO 6105 DATA SCIENCE ENGINEERING METHODS AND TOOLS

PRESENTER: SRUSHTI DHAMANGAONKAR

PROFESSOR: RAMKUMAR HARIHARAN

DATE: 12TH AUG'2020

“The Future of Search will be about pictures rather than Keywords”

BEN SILBERMANN – CEO, PINTEREST

Natural Images

Natural images Dataset, which contains 6,899 images in 8 different classes ***airplane, car, cat, dog, flower, fruit, motorbike*** *and* ***person.***

The motive of this project is to identify the above images using Neural Network.

The dataset is available at Kaggle.

Convolutional Neural Network is used here to train the Natural Images dataset and to identify images out of the given 8 category.

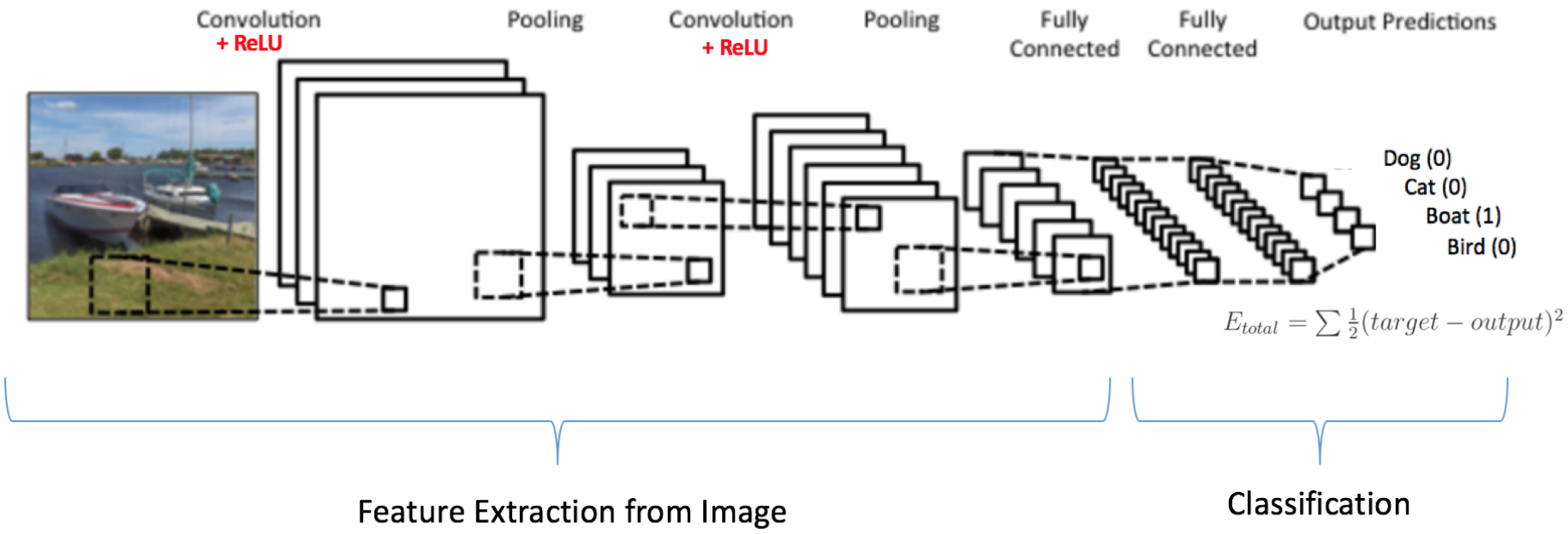
Convolutional Neural Networks (CNN)

What is CNN?

*A **convolutional neural network (CNN)** consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of **convolutional layers** that convolve with a multiplication or other dot product.*

*The activation function is commonly a Rectified Linear Unit (RELU) layer and is subsequently followed by additional convolutions such as **pooling layers, fully connected layers** and **normalization layers**, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.*

CNN Model Illustration



Convolutional Layers

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume.

The input to CNN is a tensor with shape (number of images, (image width , image height), image depth).

```
classifier = Sequential()
classifier.add(Convolution2D(32, (3, 3), input_shape = (128, 128,
  3), activation = 'relu', padding='same'))
```

Convolution layer id defined as sequential, Sequential model is nothing but a linear stack of layers.

.add() takes each layer details and stacks one after the other.

Convolution2D is Convolution operator for filtering windows of two-dimensional inputs which takes, in this case, input_shape = (128, 128, 3) along with activation function like ‘ReLu’ or ‘Softmax’. Along with these parameters we pass padding as ‘same’ (padding for image border) or ’valid’ (no padding for image border)

Pooling

Pooling layers *reduce the dimensions of the data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2 x 2.*

Max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Average pooling uses the average value from each of a cluster of neurons at the prior layer.

```
classifier.add(MaxPooling2D(pool_size = (2, 2)))
```

Fully Connected Layers

Fully connected layers *connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.*

We can say, In a ***fully connected layer****, the receptive field is the entire previous layer.*

```
classifier.add(Dense( 8, activation = 'softmax'))
```

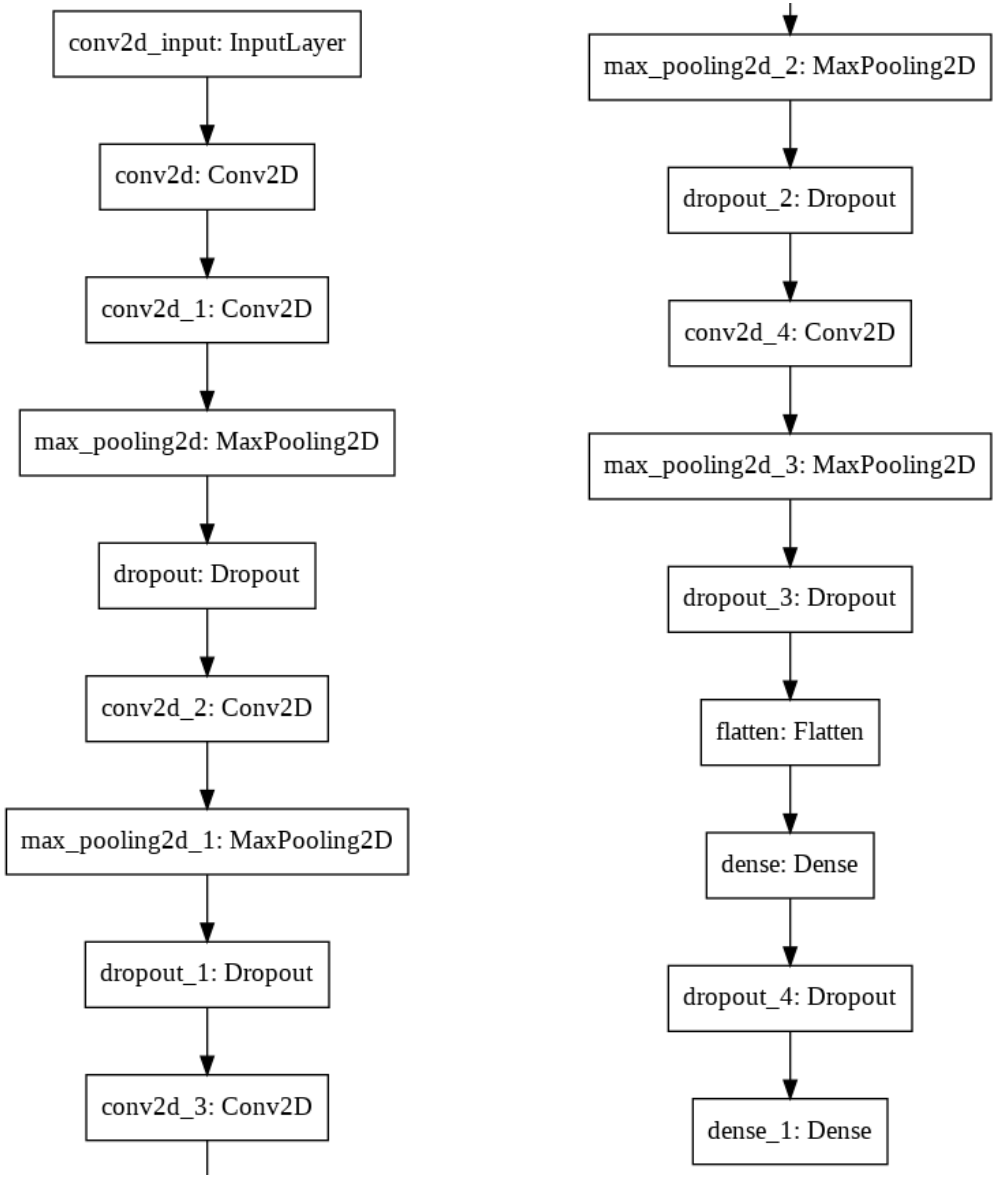
In a ***convolutional layer****, the receptive area is smaller than the entire previous layer.*

CNN Sequence Plot Model

Sequence of the model layer implemented in real time

Notice that we have a layer that randomly drops 25% of the features to prevent ***overfitting****.*

The Code base for the Model is given in the next slide.



```
# Initialising the CNN
classifier = Sequential()
# Step 1 - Convolution
classifier.add(Convolution2D(32, (3, 3), input_shape = (128, 128, 3), activation = 'relu',padding='same'))
# Adding a second convolutional layer
classifier.add(Convolution2D(32, (3, 3), activation = 'relu',padding='same'))
# step 2 - Pooling layer
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Dropout(0.25))
classifier.add(Convolution2D(64, (3, 3), activation = 'relu',padding='same'))
classifier.add(Convolution2D(64, (3, 3), activation = 'relu',padding='same'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Dropout(0.25))
classifier.add(Convolution2D(128, (3, 3), activation = 'relu',padding='same'))
classifier.add(Convolution2D(128, (3, 3), activation = 'relu',padding='same'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Dropout(0.25))
classifier.add(Convolution2D(256, (3, 3), activation = 'relu',padding='same'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Dropout(0.25))
# Step 3 - Flattening
classifier.add(Flatten())
classifier.add(Dense(1024,activation='relu'))
classifier.add(Dropout(0.5))
# Step 4 - Full connection
classifier.add(Dense( 8, activation = 'softmax'))
```

Counteracting Overfitting:

Data Augmentation and Dropping Layers

Data Augmentation: *One way to mitigate the overfitting problem is to perform data augmentation by making random transformations of the training images; The data augmentation parameters used in this model are:*

```
ImageDataGenerator(zoom_range=0.2, horizontal_flip=True, shear_range=0.2,
width_shift_range=0.2, height_shift_range=0.2)
```

Dropping Layers: *The convolution and pooling layers generate lots of feature maps from the training images. Randomly dropping some of these feature maps helps vary the features that are extracted in each batch, ensuring the model doesn't become overly-reliant on any one dominant feature in the training data.*

```
classifier.add(Dropout(0.25))
```

Model Accuracy on Test Data

Model without Data Augmentation

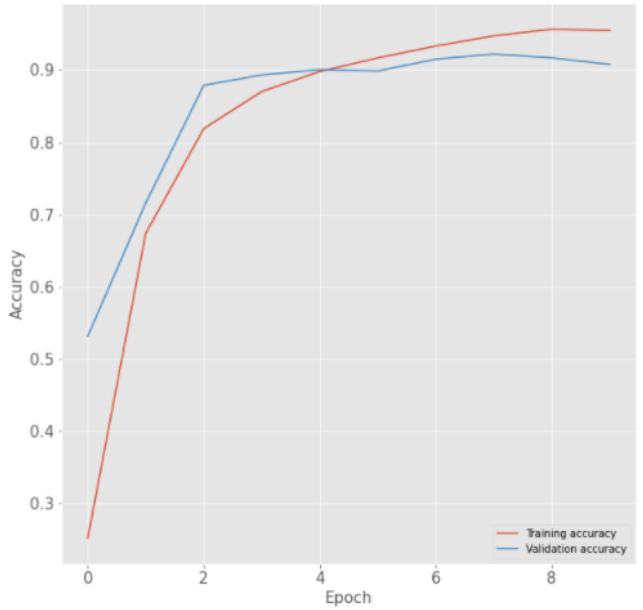
- Total test data: 1380
- Accurately predicted data: 1246
- Wrongly predicted data: 134
- Accuracy: 90.29 %

Model with Data Augmentation

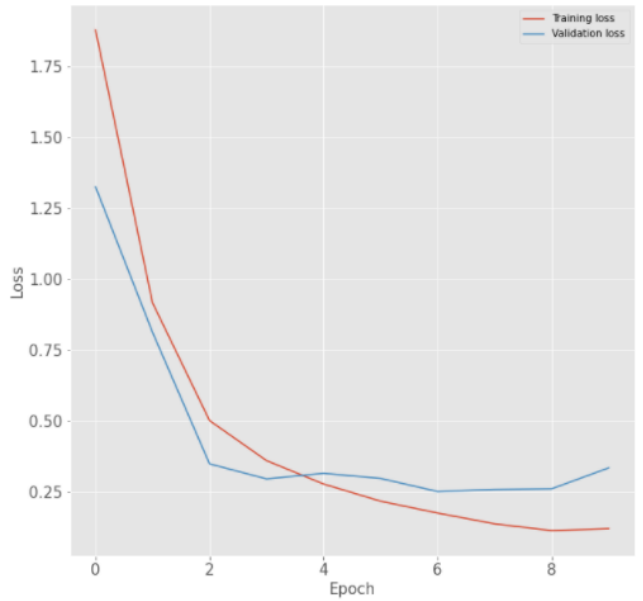
- Total test data: 1380
- Accurately predicted data: 1263
- Wrongly predicted data: 117
- Accuracy: 91.522 %

Model Accuracy and Loss Graphs

ACCURACY VS EPOCH



LOSS VS EPOCH



Conclusion

We Studied the CNN – Deep Neural Network by implementing and building the CNN model to Train and Test our model successfully on the Kaggle Natural Images Dataset.

The Model was able to get an accuracy in range 90 – 92%. We enhanced the Model accuracy by implementing Image Data Augmentation in this model.

This Project help me in understanding the real-world application of Data Science Engineering Methods and Tools.

The model can be enhanced by using VGG16 (A CNN architecture) and by also giving a larger dataset from ImageNet.