Why CNN for Image Classification?

Using an ANN for the purpose of image classification would end up being very costly in terms of computation since the trainable parameters become extremely large.

For example, if we have a 50 X 50 image of a cat, and we want to train our traditional ANN on that image to classify it into a dog or a cat the trainable parameters become –  
(50\*50) \* 100 image pixels multiplied by hidden layer + 100 bias + 2 \* 100 output neurons + 2 bias = 2,50,302.

Convolutional Neural Networks come under the subdomain of Machine Learning which is Deep Learning. It was proposed by computer scientist ***Yann LeCun*** in the late 90s, when he was inspired from the human visual perception of recognizing things.

Construct CNN convolution layer follows a hierarchical model which works on building a network, like a funnel, and finally gives out a fully-connected layer where all the neurons are connected to each other and the output is processed.



A convolution network is a multilayer feedforward network that has two- or three-dimensional inputs. It has weight functions that are not generally viewed as matrix multiplication (or inner product) operations, The principal layer type for convolution networks is the

convolution layer.

Diagram

Description automatically generated

**TensorFlow:**

When it comes to picking a framework for your deep learning project, people are always debating between TensorFlow and PyTorch. Both frameworks are very useful abstractions and reduce a decent amount of code and speed up model development.

 In this project we choose to use TensorFlow however we feel It will fits our needs and it has the most features we need to develop our model in a simple way.

**Model definition and building network:**

First we will start with the CNN model we designed using TensorFlow, and then will talk about the Pertained model.

**Step 1: Data Preprocessing:**

As many of you already know working with Images fakes classification there is no straight way to use a model that fit with all types of photoshopped or fakes images so we start to apply different filters to the images and apply model to see which model work best with which type of filter to give a better results.

So we end up with 4 different class of data:

1. Class01 – Original Images with no filter
2. Class02 – Apply ELA (Error Level Analysis)
3. Class03 –

**Step 2: Prepare Dataset for Training**

|  |  |
| --- | --- |
| Create Training & Validation Data:  We divided the data into two groups: one for training and the other for model validation in a ratio of 80/20. | Text  Description automatically generated |
|  | Text  Description automatically generated |

**Step 3: Images Augmentation:**

|  |  |
| --- | --- |
| We dfine a sperate layer of Image augmentation and added later to CNN Architectures utilizing the GPU and to minimize the overfitting | Text  Description automatically generated with low confidence |

**Step 4: Scheduler:**

|  |  |
| --- | --- |
| Create a Scheduler function that help to dynamically assign Learning Rates while the model optimizing the weights, although some optimizer like “Adam” has built-in features to adjust the “LR” while the training process. | Text  Description automatically generated |

**Step 5: Build the convolutional neural network:**

In this step we define the convolutional neural network where the convolution, **Pooling**, and **Fattening** layers will be applied. We also added **Dropout layer** to reduce overfitting. Since it’s a binary class problem, the **Sigmoid activation** function is applied in the last layer.

Text

Description automatically generated

#### **Step 6 : Compiling the model:**

The next step is to compile the model. The Binary Crossentropy loss is used because the labels are binary. In the event that you want to encode the labels, then you will have to use the [Categorical Cross-Entropy loss](https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy) function.

Text

Description automatically generated

#### **Step 7 : Early Stopping, Model Check Point :**

Early stopping: To avoid that is to stop the training process when the model stops improving

Model Check Point : automatically saving the best model or model weights during training

Text

Description automatically generated

#### **Step 8: Training the model:**

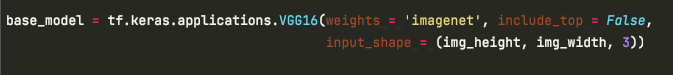
Now we are ready to fit the data to the training set. The model is set to run for 50 epochs but the process will be stopped by the callback when the loss doesn’t improve after 5 epochs as we defined in the Early Stopping callback.

Text

Description automatically generated

**Pretrained Model:**

In this project we choose to use **VGG16** model using Keras API. This model and others like (Xception, ResNet50 & MobileNet) have been pre-trained on the ImageNet dataset which has over million images.



Show the deep learning model results:

Text

Description automatically generated