LAB ASSIGNMENT-5

Advance CNN Architectures: LeNet, AlexNet
For Stanford dogs dataset

BY,

MADASU DEEPIKA
19MIA1066

Abstract:

Dog breed categorization is a very specific application of convolutional neural networks. It falls under the category of fine-grained image classification problem, where inter-class variations are small and often one small part of the image considered makes the difference in the classification. The various classes of ImageNet can have large inter-class variations, making it easier to categorize correctly. In this work, I aim to use a convolutional neural network framework to train and categorize dog breeds using CNNs based on LeNet and Alexnet.

Keywords: Dogs, Deep Learning, LeNet, Convolution Neural Network.

Introduction:

Convolutional neural networks (CNN) have been used to great effect in applications such as object classification, scene recognition, and other applications.

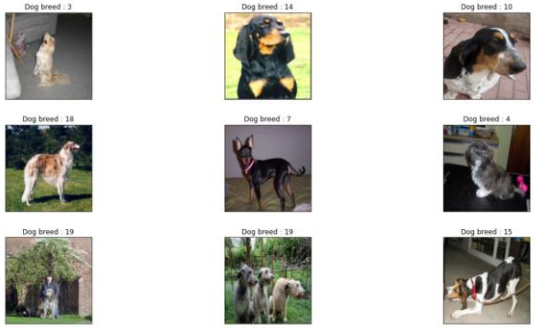
In many situations, we can imagine the features (both low-level and higher-level) that are learned by the CNNs in the process of training. However, when the objects the CNNs are trying to categorize share many similar features, such as the breeds of dogs, it becomes hard to imagine the specific features that CNNs must learn in order to categorize these dogs correctly.

It is therefore interesting to see how well CNNs can perform on only dog breeds, compared to labels from all classes of objects in the regular ImageNet.

Dataset:

The Stanford Dogs dataset is an open-access image dataset of dog breeds. There are a total of 120 classes of dogs, with 20580 images in total, partitioned into 8580 test images, and 12000 training images. The image dataset comes with annotations that mark out the bounding boxes which best include the dogs in the image.
This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization.
It was originally collected for fine-grain image categorization, a challenging problem as certain dog breeds have near identical features or differ in colour and age.

Dataset link: [Stanford Dogs Dataset | Kaggle]



Methodology:

We need to split our dataset into three parts: training dataset, test dataset, and validation dataset. The purpose of splitting data is to avoid overfitting which is paying attention to minor details/noise which is not necessary and only optimizes the training dataset accuracy. The training set is the actual subset of the dataset that we use to train the model. The model observes and learns from this data and then optimizes its parameters.

The test set is there a remaining subset of data used to provide an unbiased evaluation of a final model fit on the training dataset. Data is split as per a split ratio which is highly dependent on the type of model we are building and the dataset itself.

In our approach, we have dedicated 80% of the dataset as the training data and the remaining 20%as the testing data, which makes the split ratio as 0.8:0.2 of the train

to test set. Out of the training data, we have used 20% as a validation data set.

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. By using the sequential method, the architecture of the model is built layer by layer, convolution layer, maxpool layer, flatten layer, drop out, dense layer.

Here are the types of layers the AlexNet CNN architecture is composed of, along with

Convolutional layer: A convolution is a mathematical term that describes a dot product multiplication between two sets of elements. Within deep learning the convolution operation acts on the filters/kernels and image data array within the convolutional layer. Therefore a convolutional layer is simply a layer the houses the convolution operation that occurs between the filters and the images passed through a convolutional neural network.

Batch Normalisation layer: Batch Normalization is a technique that mitigates the effect of unstable gradients within a neural network through the introduction of an additional layer that performs operations on the inputs from the previous layer. The operations standardize and normalize the input values, after that the input values are transformed through scaling and shifting operations.

MaxPooling layer: Max pooling is a variant of sub-sampling where the maximum pixel value of pixels that fall within the receptive field of a unit within a sub-sampling layer is taken as the output. The max-pooling operation below has a window of 2x2 and slides across the input data, outputting an average of the pixels within the receptive field of the kernel.

Flatten layer: Takes an input shape and flattens the input image data into a one-dimensional array.

Dense Layer: A dense layer has an embedded number of arbitrary units/neurons within. Each neuron is a perceptron.

Some other operations and techniques utilized within the AlexNet CNN that are worth mentioning are:

Activation Function: A mathematical operation that transforms the result or signals of neurons into a normalized output. The purpose of an activation function as a component of a neural network is to introduce non-linearity within the network. The inclusion of an activation function enables the neural network to have greater representational power and solve complex functions.

Rectified Linear Unit Activation **Function(ReLU)**: A type of activation function that transforms the value results of a neuron. The transformation imposed by ReLU on values from a neuron is represented by the formula y=max(0,x). The ReLU activation function clamps down any negative values from the neuron to 0, and positive values remain unchanged. The result of this mathematical transformation is utilized as the output of the current layer and used as input to a consecutive layer within a neural network.

Softmax Activation Function: A type of activation function that is utilized to derive the probability distribution of a set of numbers within an input vector. The output of a softmax activation function is a vector in which its set of values represents the probability of an occurrence of a class or event. The values within the vector all add up to 1.

Dropout: Dropout technique works by randomly reducing the number of interconnecting neurons within a neural network. At every training step, each neuron has a chance of being left out, or rather, dropped out of the collated contributions from connected neurons.

The code snippet represents the Keras implementation of the AlexNet CNN architecture.

AlexNet Cnn Model:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 55, 55, 96)        34944

 max_pooling2d_6 (MaxPooling  (None, 27, 27, 96)       0
 2D)

 batch_normalization_12 (Bat  (None, 27, 27, 96)       384
 chNormalization)

 conv2d_11 (Conv2D)          (None, 23, 23, 256)       614656

 max_pooling2d_7 (MaxPooling  (None, 11, 11, 256)      0
 2D)

 batch_normalization_13 (Bat  (None, 11, 11, 256)      1024
 chNormalization)

 conv2d_12 (Conv2D)          (None, 9, 9, 384)         885120

 conv2d_13 (Conv2D)          (None, 7, 7, 384)         1327488

 conv2d_14 (Conv2D)          (None, 5, 5, 256)         884992

 max_pooling2d_8 (MaxPooling  (None, 2, 2, 256)        0
 2D)

 batch_normalization_14 (Bat  (None, 2, 2, 256)        1024
 chNormalization)

 flatten_2 (Flatten)         (None, 1024)              0

 dense_8 (Dense)             (None, 4096)              4198400

 batch_normalization_15 (Bat  (None, 4096)             16384
 chNormalization)

 dense_10 (Dense)            (None, 1000)              4097000

 batch_normalization_17 (Bat  (None, 1000)             4000
 chNormalization)

 dense_11 (Dense)            (None, 20)                20020

=================================================================
Total params: 28,883,132
Trainable params: 28,863,532
Non-trainable params: 19,600
_________________________________________________________________
```

Figure 1

LeNet Cnn Model:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 224, 224, 32)      2432

 max_pooling2d_6 (MaxPooling  (None, 112, 112, 32)     0
 2D)

 conv2d_11 (Conv2D)          (None, 108, 108, 48)      38448

 max_pooling2d_7 (MaxPooling  (None, 54, 54, 48)       0
 2D)

 flatten_2 (Flatten)         (None, 139968)            0

 dense_6 (Dense)             (None, 256)               35832064

 dense_7 (Dense)             (None, 84)                21588

 dense_8 (Dense)             (None, 20)                1700

=================================================================
Total params: 35,896,232
Trainable params: 35,896,232
Non-trainable params: 0
_________________________________________________________________
```

Figure 2

Result:

AlexNet CNN model:

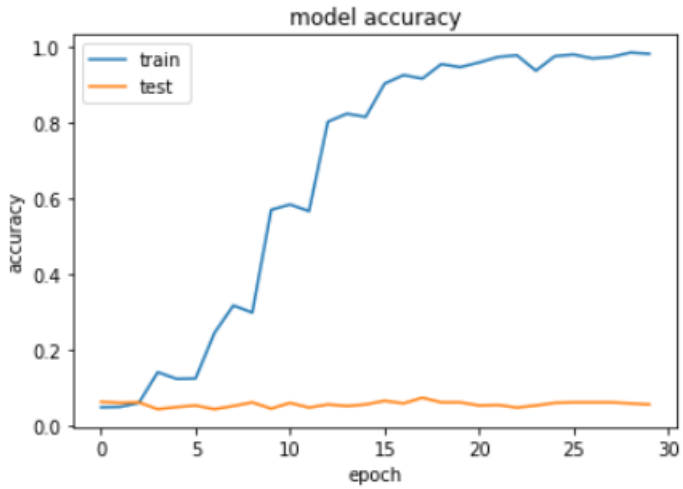
1) **Epoch=5**

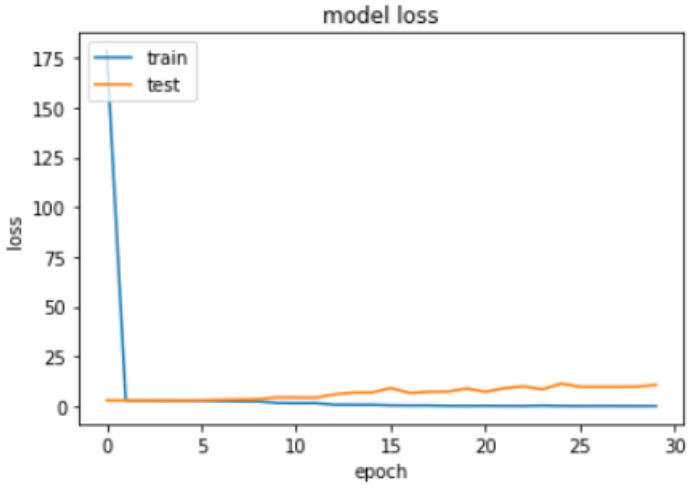
```
Epoch 1/5
91/91 [==============================] - 122s 1s/step - loss: 3.3394 - accuracy: 0.1149
Epoch 2/5
91/91 [==============================] - 104s 1s/step - loss: 2.8705 - accuracy: 0.1549
Epoch 3/5
91/91 [==============================] - 98s 1s/step - loss: 2.7422 - accuracy: 0.1807
Epoch 4/5
91/91 [==============================] - 97s 1s/step - loss: 2.6950 - accuracy: 0.1869
Epoch 5/5
91/91 [==============================] - 105s 1s/step - loss: 2.6709 - accuracy: 0.1948
```

2) **Epoch=10**

```
Epoch 1/10
91/91 [==============================] - 100s 1s/step - loss: 2.5926 - accuracy: 0.2261
Epoch 2/10
91/91 [==============================] - 114s 1s/step - loss: 2.5496 - accuracy: 0.2275
Epoch 3/10
91/91 [==============================] - 103s 1s/step - loss: 2.4940 - accuracy: 0.2440
Epoch 4/10
91/91 [==============================] - 95s 1s/step - loss: 2.4019 - accuracy: 0.2546
Epoch 5/10
91/91 [==============================] - 98s 1s/step - loss: 2.3332 - accuracy: 0.2856
Epoch 6/10
91/91 [==============================] - 97s 1s/step - loss: 2.3212 - accuracy: 0.2863
Epoch 7/10
91/91 [==============================] - 97s 1s/step - loss: 2.2037 - accuracy: 0.3138
Epoch 8/10
91/91 [==============================] - 98s 1s/step - loss: 2.0757 - accuracy: 0.3510
Epoch 9/10
91/91 [==============================] - 98s 1s/step - loss: 2.0331 - accuracy: 0.3747
Epoch 10/10
91/91 [==============================] - 105s 1s/step - loss: 1.9356 - accuracy: 0.3981
```

Training and Validation loss & accuracy of LeNet model:

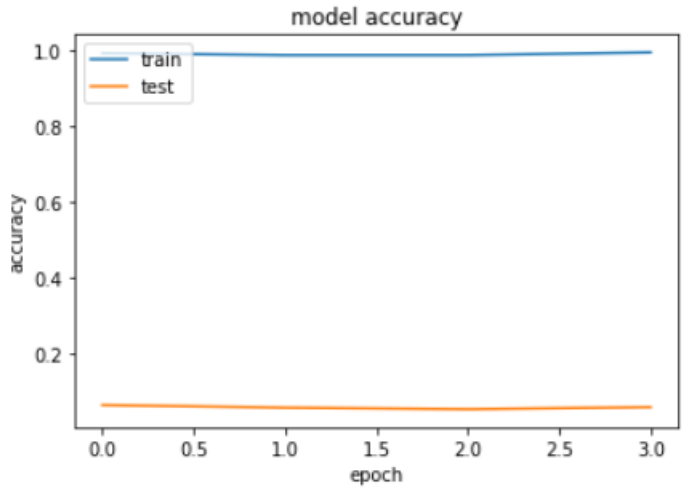


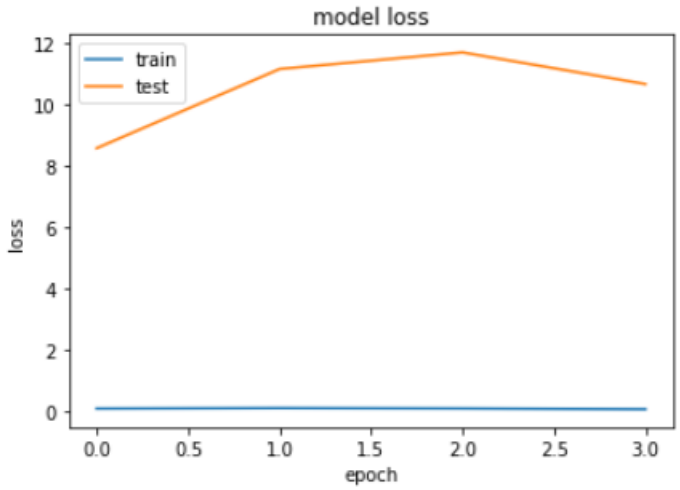


```
Final training loss      8.947017788887024
Final training accuracy  98.22916388511658
```

As the model trains, the final training loss and accuracy metrics are displayed. This model reaches an accuracy of about 98.22% on the training data and the loss of 8.94%.

Training and Validation loss & accuracy of LeNet model Using Early Stopping:






```
Final training loss      4.4443778693675995
Final training accuracy  99.37106966972351
```

As the model trains, the final training loss and accuracy metrics are displayed. This model reaches an accuracy of about 99.37% on the training data and the loss of 4.443%.

Conclusion:

In this paper, we implemented an application using the Deep Convolutional Neural Network Advanced Algorithm model. To understood the key concepts of advanced architectures in CNN using Alexnet and lenet model.

References:

fcdh_FinalReport.pdf (stanford.edu)

(PDF) Dog Breed Identification Using Deep Learning (researchgate.net)

Stanford Dogs dataset for Fine-Grained Visual Categorization

CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more - coderz.py (coderzpy.com)

CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more… | by Siddharth Das | Analytics Vidhya | Medium

7.1. Deep Convolutional Neural Networks (AlexNet) — Dive into Deep Learning 0.17.4 documentation (d2l.ai)

Implementing AlexNet CNN Architecture Using TensorFlow 2.0+ and Keras | by Richmond Alake | Towards Data Science