"Image Classification using Convolutional Neural Networks"

Mundhe Prathmesh mundhe.p@husky.neu.edu
Walimbe Pranav walimbe.p@husky.neu.edu
Kulkarni Sumedh kulkarni.sum@husky.neu.edu

ABSTRACT

In this project, we plan to develop an algorithm for classifying images of 101 different classes like 'helicopter', 'airplanes' and 'faces'. We investigated the deep neural networks to address this problem. We used the Convolutional Neural networks (CNN) to extract and learn features of the images and train our model for classification. We tried various experiments to improve the accuracy on our test dataset and finally achieved an accuracy of 61.13% by this approach. We also compared the impact of two different activation functions on our model and Rectifier (relu) activation function gave us better accuracy and performance as compared to tanh activation function.

General Terms

Convolutional Neural Networks, Prediction, Classification.

Keywords

Classification, CNN, ANN, relu, tanh.

INTRODUCTION

1.1 Overview

The rise of big data and popularizing of high computing devices have contributed to the development of machine learning. Image classification is one of the important topics in the field of machine learning. This project involves artificial extraction of features using the Sequential model and training the model to classify and predict the class of a new input image. We plan to compare the performance of two models using two different activation function: 'relu' and 'tanh'.

1.2 Motivation

Image classification based on their classes is easy for humans but difficult for machines. We plan to train a model on many images from various classes to build a model for prediction.

We want to compare results for two linear activation functions: 'tanh' and 'relu' and plot the differences in accuracy and loss for the model.

1.3 Approach

Out basic task is to create an algorithm to classify image into the dataset classes. We have used the Convolutional neural networks algorithm for training the model using keras.

We have compared the results of two models using two different activation functions and used accuracy and loss function to evaluate the performance of the model.

We have also added dropout to reduce the overfitting of the model. We have developed two different models to compare the results of two different activation functions.

1.4 Dataset

We have used the Caltect 101 dataset. It is a public dataset available from the California Institute of technology machine learning repository and it can be downloaded from [1].

The dataset contains images belonging to 101 classes. We have split the dataset in 80% training set and 20% test set and have used for generating results by feature extraction and labels.

BACKGROUND

Many projects have been previously done in the field of Image classification using CNN. We student various previous research papers and learned to modify our model to increase accuracy and performance.

We started by using the 101 classes of the dataset and achieved 29% accuracy by using a single layer CNN. **Alex Krizhevsky** and **Ilya Sutskever** [2] published a paper to suggest using multiple layers and adding regularizers to improve accuracy.

Also, adding dropout would help in reducing overfitting and improving accuracy.

We then applied dropout to our model and the accuracy of the model increased to 61.13%.

But in order to increase the accuracy of the model, we decided to reduce the scope of the project and select 10 classes to develop a predictive model on. Hence reducing the number of classes helped us in increasing the performance of the model.

We also wanted to compare the performance of model under two different activation functions. We have used relu and tanh as our activation functions for prediction models.

APPROACH

3.1 Overview

- Image class analysis
- Understanding the image structure and neural networks theory
- CNN (Python)
- Development of algorithms for image classification
- Comparison of models using different activation functions
- Predictive Model using CNN

3.2 Route

3.2.1 Neural Networks

The basic entity of a neural network is a perceptron. There is an input layer which provides to this perceptron. This perceptron gives us the output.

The input in our case is images. These images are split into training and testing dataset. The model then trained over the training dataset. Then this model is used to test with the testing dataset. In the case of images, the input data is rescaled and normalized. The last part of the process is to predict the class of a random input data

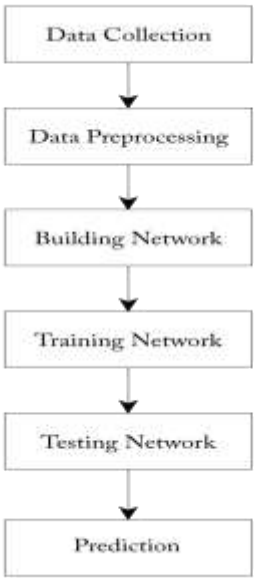


Fig. 3.1 – Modelling a Neural Network

3.2.2 Layer types and description of Convolutional Neural Networks

3.2.2.1 Image Normalization

The first step in our approach is to normalize the input image. Images are given as a set of RGB values. These values are given as channels. Value of 1 equals grey scale and of 3 equals that of RGB. So, we need to normalize these values in a common range. Hence in the first layer we have normalize our images to RGB using scale 3.

3.2.2.2 Rectified Linear Unit

We use ReLU to threshold the input. These are usually defined in the convolutional layer itself. In tensorflow there are two ways to define it. First to define it in the convolutional layer and secondly add a different layer that contains the activation function.

We use the first option in our approach. The formula for rectified linear unit is given as:

$f(x) = max(0, x)$

3.2.2.3 Convolutional Layer

The convolutional layer convolves a set of filters over the input. A high filter response indicates similarity between the filter and the input and vice-versa. The filter outputs obtained in this layer enables it to make a decision about the class of the input image. It does linear transformation from input to output without changing the dimensions of the input image but changing the number of channels in the output image. The convolutional layer takes the input with some pre-defined *bias*. *Weights* are defined for each input as well. These weights help us reduce the errors when we backpropagate the error value and adjust it so as to improve the model.

3.2.2.3 Maxpool layer

The maxpool operation is used to down-sample the images. This takes in an input of size and turns it into our desired size. In this step we change the number of rows and columns but the depth remains same. The maxpool operation is important because it will not overfit to our data.

3.2.2.4 Fully connected layer

This layer comes into use when we are done with all our convolutional, maxpool and ReLU operations. This layer computes the numerical values of the given input, which is the output of our above operations.

3.2.2.5 Soft-max Layer

The end of a convolutional neural network is usually either a softmax classifier or SVM. In our network we have used softmax. The softmax classifier takes in an array and gives output for different categories in the dataset.

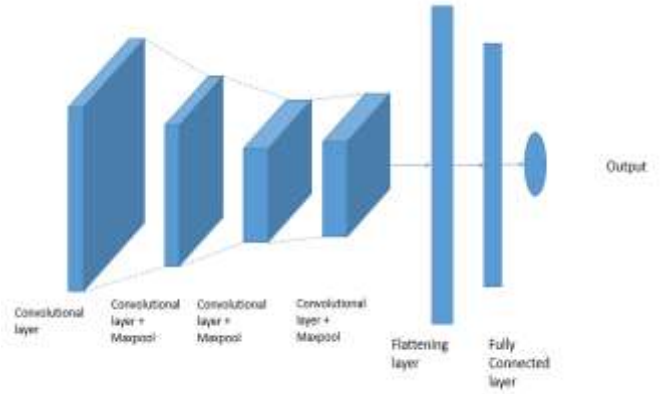


Fig. 3.2 Convolutional Neural Network

METHODOLOGY

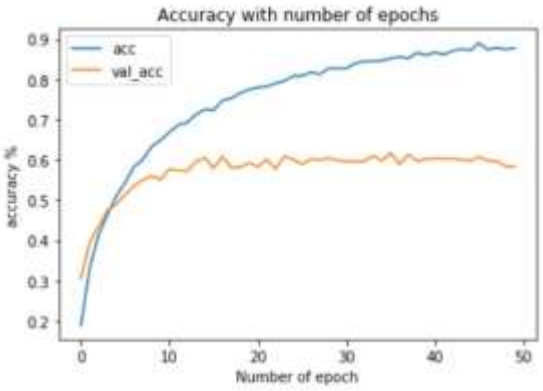
1. The dataset was downloaded from caltech's website. There were 101 classes which contained multiple images. We then split the dataset into 80% training and 20% testing. The model was designed using tensorflow in keras with a combination of convolutional and max pooling layers. This was then flattened and provided to the fully connected layer. The validation accuracy that we got was around 30%. With some tweaking of the hyper parameters we were able to get accuracy of up to 62%.
2. This was happening due to class imbalance. So we decided to restrict our training to 10 classes which had maximum number of images.
3. As our model was ready, we had a lot of space to improve our model's performance on 10 classes. We decided to go ahead with 2 activation functions and then compare the results. The two activation functions were, 'tanh' and 'relu'. As an initiative to learn the working of the neural network, we decided to visualize the output of every convolutional and max pooling layer. Using tanh activation we were able to get 32% accuracy.
4. We then used relu activation function for the same model. With relu activation function, we were able to get 93% accuracy for our model.
5. We even plotted multiple graphs throughout the progress of the project. We compared the training accuracy vs validation accuracy for all the models, training loss vs validation loss for all the models. We even plotted the comparison plots for tanh and relu activation functions.

RESULTS AND ANALYSIS

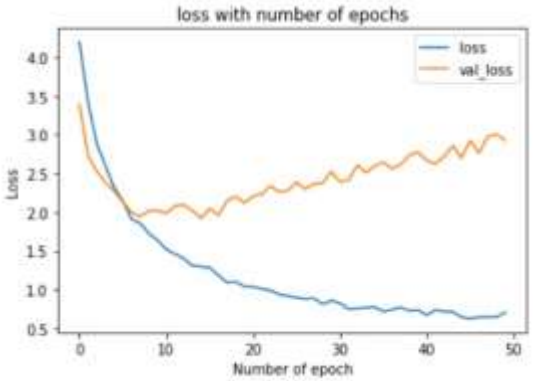
- At the end of the project we have thorough understanding of CNN and its application for image classification.
- Designed different models for comparisons and performance evaluation.
- Successfully loaded the dataset into training and designed a model for predicting the class of a new test image.
- Evaluated the performance of model under two different activation functions – 'relu' and 'tanh' and achieved 92% accuracy using relu activation function and 31% accuracy using tanh activation function, clearly stating the impact of activation function on the performance of a model.
- Also, designed a model for all 101 classes of the dataset and achieved 61% accuracy.

Results for 101 categories:

1. Plot for training accuracy vs validation accuracy

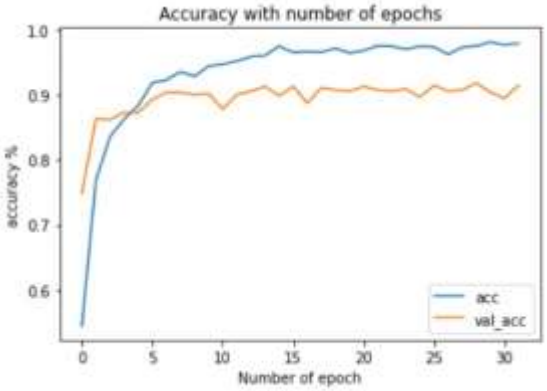


2. Plot for training loss vs validation loss

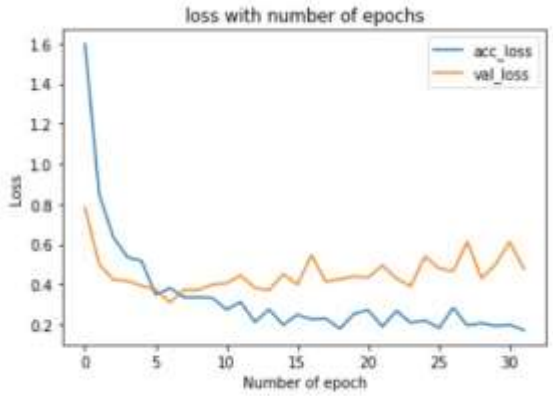


Results for 10 categories:

1. Plot for training accuracy vs validation accuracy

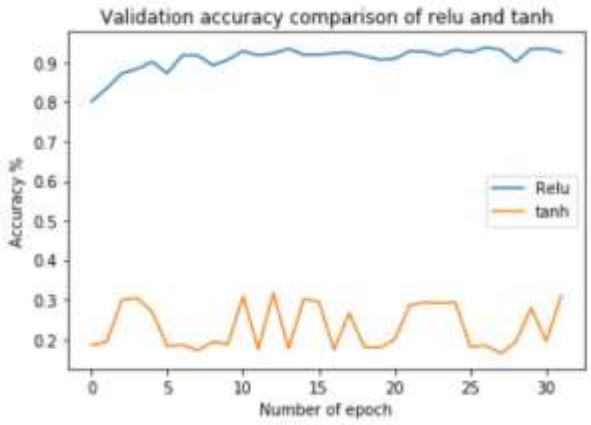


2. Plot for training loss vs validation loss

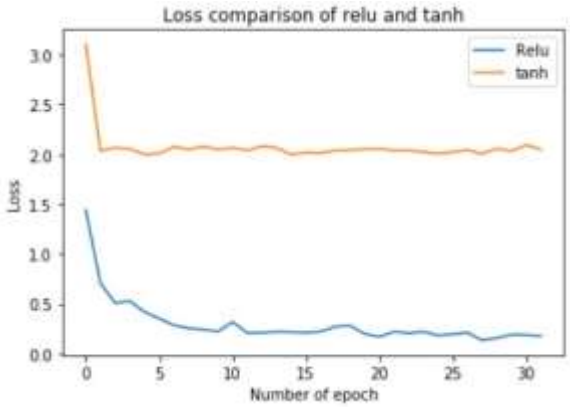


Results for comparison between relu and tanh activation function:

1. Plot for validation accuracy between two models



2. Plot for loss between two models



Comparison of validation accuracy across models

	Accuracy	Validation Acc
For 101 classes	95.25	61.13
For 10 classes	98.11	91.80
For tanh activation function	30.95	31.69

CONCLUSION

- Relu activation function performed better as compared to tanh for our model.
- Imbalance between classes in a dataset gives undesired results and low accuracy as seen by 61% accuracy with 101 classes.
- Increasing the number of convolution layers in a fully connected model helped us increase accuracy.

ACKNOWLEDGEMENT

We would like to thank our Professor Jerome Braun and our Teaching assistant Ramin Mohammadi for their extraordinary support and guidance throughout the entire project.

REFERENCES

[1] http://www.vision.caltech.edu/Image_Datasets/Caltech101/

[2] “ImageNet Classification with Deep Convolutional Neural Networks.” Alex Krizhevsky and Ilya Sutskever.

[3] http://luthuli.cs.uiuc.edu/~daf/CV2E-site/classificationextracts.pdf