# QSTP

# Introduction to Deep Learning

## Research and Analysis

### Computer Vision: Image Classification

SUBHRAJYOTI DUTTA

Aug 2021

1. Introduction

   In today's world of robots, self-driving cars and artificial intelligence, Computer Vision has come up to be one of the most important gears. It is helping in facial recognition, predicting obstacles in path, etc.

   We will be building a model which will help us in recognising the object present in an image. We will be using the STl 10 dataset provided by Stanford University and curated by Prof Adam Coates, Prof Honglak Lee and Prof Andrew Ng.

   The given dataset has 10 categories, namely: airplane, bird, car, cat, deer, dog, horse, monkey, ship and truck.

   Our goal is to make a model which can identify features and thus categorise the picture.

2. Literature Review

   We already know about logistic regression and linear regression. In Linear regression we tend to predict the value of outcome based on the given factors or conditions. In Logistic regression we use it for binary classification purposes. Its output is a probability showing the chances / possibility of a new item to fall in one or the other category.

   We also know that for multi-class classification, we tend to use multiple logistic regression models on the same data to categorize them.

3. Hypothesis

   We will use multiple layers and we will use the output of one layer to be the input of the next layer while using multiple logistic regressions on each layer. Doing this will help us extract new features / components which will further help in determining the outcome. We will call it a Neural Network.

   Also we will create a few layers at the beginning in such a way that it will extract certain specific features from the previous layer, making it easier for us to draw information from it. We will call this type of neural network where there is specific feature extracting as Convolutional Neural Network.

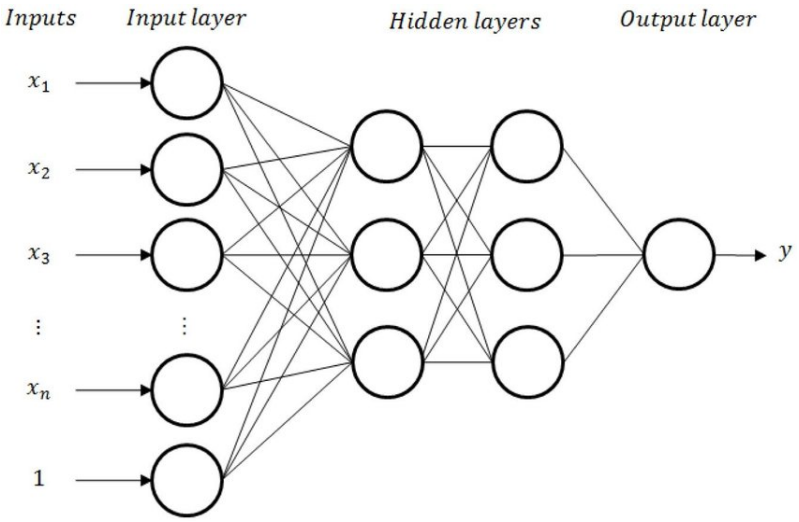



Fig 1: Neural Network

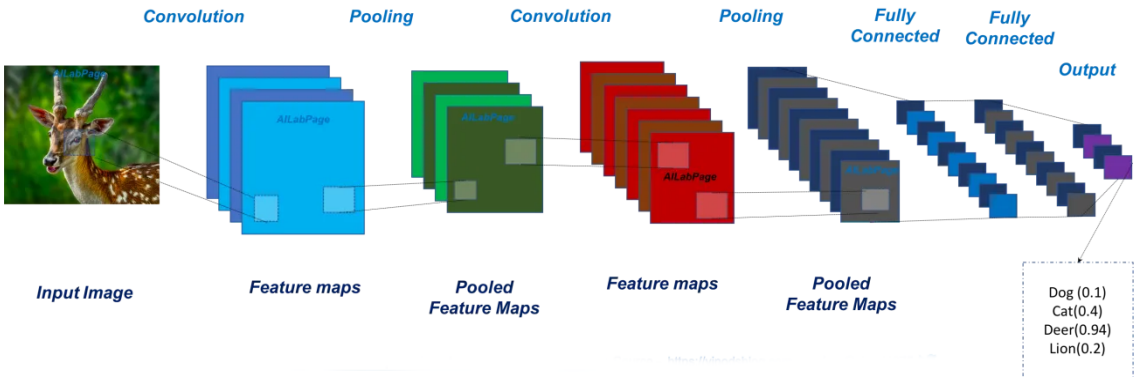


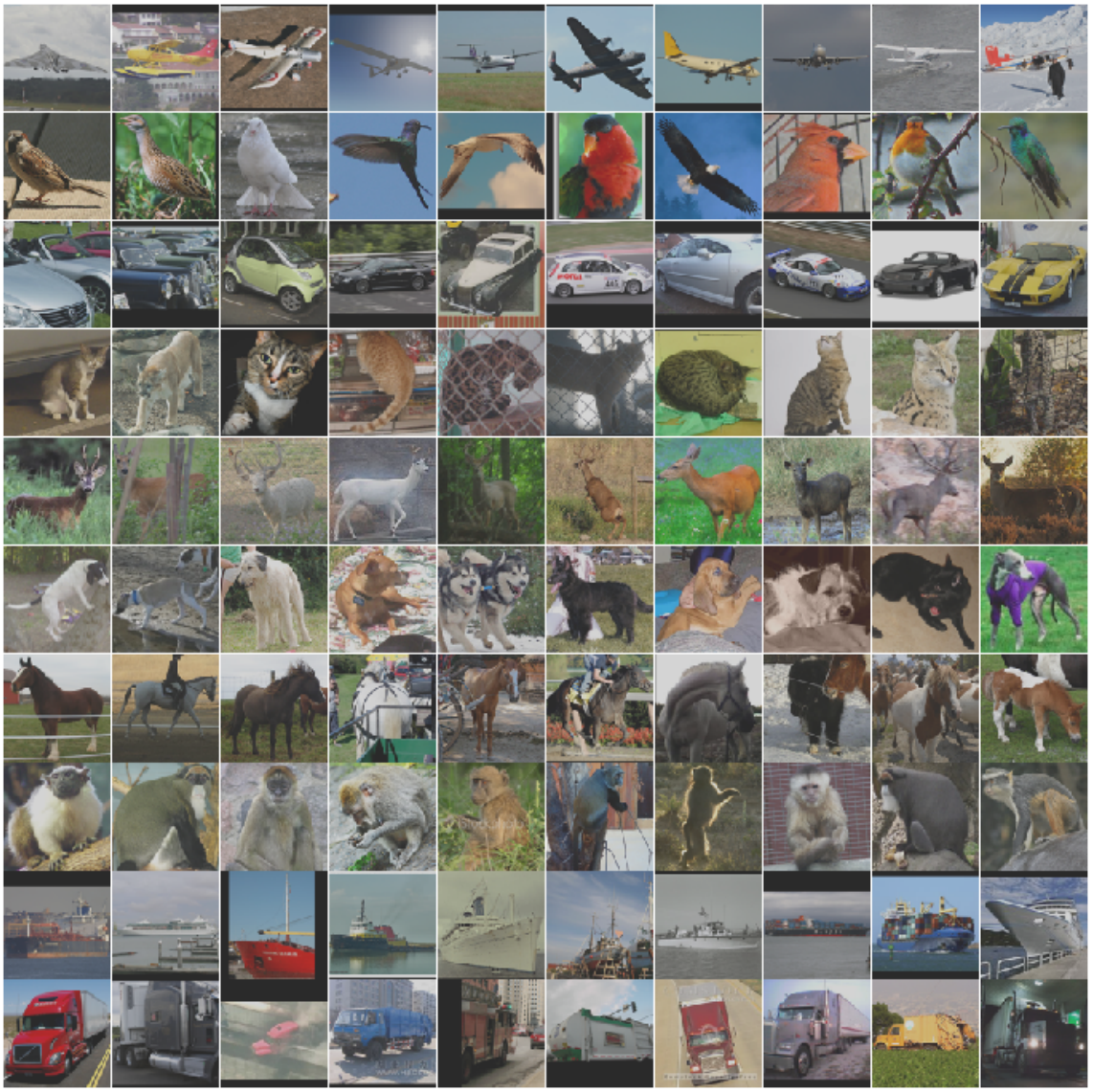

Fig 2: Convolutional Neural Network (CNN)

Like as we used the loss in the logistic regression to tune our logistic function, we will use the loss value in CNN and use it to tune each and every coefficient in the CNN. We will call this back-propagation.

4. Material

For this there are multiple frameworks we can use, like: sklearn, torch, tensorflow, theanos etc. But for our purpose, we will be using tensorflow, developed by Google developers. It is fast, smooth and easily scalable. It is one of the best machine learning libraries currently present.

We will be executing the model on the STL 10 dataset. It consists of 5000 training images (500 from each category) and 8000 test images (800 from each category) along with 100000 unlabelled images.



5. Planning and drafting

   I had to decide on the number of layers and number of nodes in each layer.

   a. For Convolutional part, after frequent trials I settled down to two convolutional layers along with 2 max pooling layers. Going higher than that had little to no effect on the outcome, whereas going lower than that had very poor results.

      Also 64 kernels were used which seemed to be a sweet number along with kernel size as 2x2.

   b. For the Neural Network part, after multiple hits and trials we settled upon 1 flattening layer and 2 fully connected hidden layers and 1 output layer. Going higher than 2 hidden layers was leading to overfitting and poor performance on test set.

      Also I settled upon 24X24 = 576 nodes per hidden layer. Going with this configuration had the best result.

6. Model Testing and Implementation

   For model testing, we are using early stopping for the purpose of regularisation. I did the early stopping based on the model performance on the validation set. The criterias used for early stopping were:

   a. Checking the validation loss if it has stopped decreasing.
      Reason: when the validation loss starts increasing it indicates that the performance of the model is decreasing.
   b. Checking if the validation loss is within 0.001 units of the previous loss value.
      Reason: if the decrement in validation loss is extremely small then it implies that the improvement in performance is negligible and this will continue henceforth.

   After testing it on the test dataset we fine turned the other minor parameters to get the best result.

*The result and conclusion in Computer Vision folder