



Optimization for Machine Learning Application

10.04.2023

University of Denver

Mohammad Mehdi Hosseini, Farida Far Poor

Optimization Application Project

Introduction

This project is image classification using deep learning methods. In this project we develop a light-weighted deep convolutional neural network, where the model is trained to first extract relevant features from data, and then predict labels. The optimization problem here is minimizing the error between the ground-truth labels and the predicted labels. In a feed forward process the model predicts a label, and then based on a loss function (for example Binary Cross-Entropy Loss) calculates the loss value. In the next phase it calculates the gradient of error and updates the weights of the model based on one of the optimization methods (such as Stochastic Gradient Descent or ADAM, etc.) in the opposite direction of the gradient, in a way that the model converges to the point that the error be minimum. Using CNN, we do not rely on manually created feature extractors or filters. Instead, convolutional filters train our model from raw pixel-level to final object classes (the magnificence of CNN) [3] [4].

Objectives

This project will explore a CNN application in image processing by evaluating the model architecture. The goal is to use optimization to pick optimized learning rate, minimize the error and pick the best optimization method for the purpose of updating the weights of the network. We will use Caltech 101 [1] and CIFAR-100 [2] datasets in our project. The project datasets information is shown in Table 1.

Dataset name	# Images	# Classes	# Train	# Test
Caltech 101	9 k	101	3060	6k
Cifar 100	60 k	100	50 k	10 k

Table 1: Datasets information

In general, neural networks work based on the perceptrons, where millions of them get together to make a network structure. Figure 1 shows a perceptron.

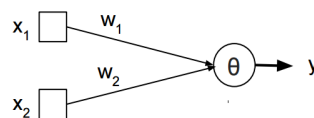
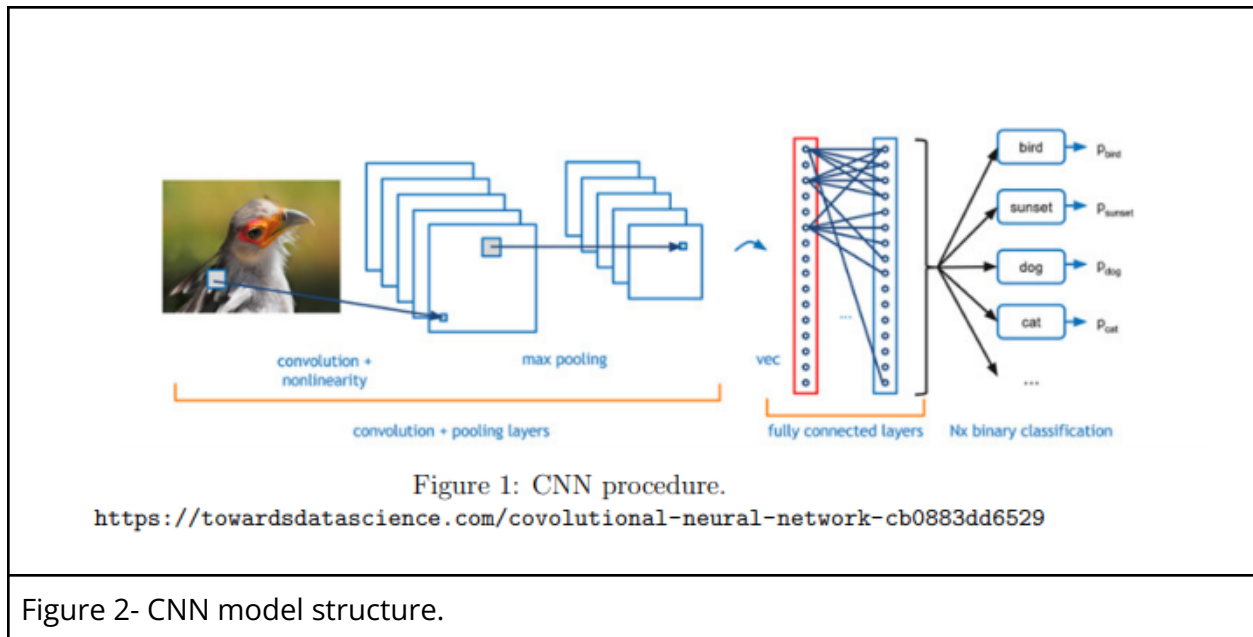


Figure 1- Perceptron structure, where x is input signal, w is the weight we are trying to learn, θ is a function, and y is the output.

Each perceptron contains two functions, the first is a sum function that integrates the input, and the second is called activation function, which is a transformation to a non-linear space. Information passes through this network, and in the final layer the error between the predicted output and the expected output is calculated and back-propagates through all the network. The figure below (Figure 2) shows the structure of a CNN network:



In this project we will study:

- The effect of different optimizers on the convergence speed and accuracy in our problem
- Choosing the appropriate learning rate
- The difference between dynamic versus constant learning rates
- At least 2 different loss functions in the classification problem (Binary Cross-Entropy, Hinge Loss, etc.)
- Two datasets Caltech-101 and Cifar-100 (each with around 100 classes) in different image sizes

We will evaluate our CNN model's performance by considering the best:

- Learning rate
- Optimizer
- Loss function

And finally, we evaluate the model accuracy by measuring

1. Precision

2. Recall
3. F1

Optimization Problem

In the context of neural networks the optimization function is related to the loss function and the constraints are learning rate range, activation function. In this project we utilize binary cross-entropy loss function, mean squared error loss function, and hinge loss function as our optimization problem. The learning rate range is a constraint, and activation is another one.

Objective functions:

- Binary Cross Entropy:

$$\min_{\theta} L(\theta) = -\frac{1}{N} \sum_{i=1}^N (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

- Hinge Loss:

$$\min_{W,b} \frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i \cdot f(x_i))$$

- Meas Squared Error:


$$\min_{\theta} L(\theta) = \frac{1}{N} \sum_{i=1}^N (Y_i - \hat{Y}_i)^2$$

Constraints:

- $0.1 < \text{learning_rate} < 0.00001$
- Number of layers
- Activation function (sigmoid, tanh, Relu, etc.)
- Optimizer (Adam, SGD)

References

- [1] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Pattern Recognition Workshop, 2004.
- [2] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

- 
- [3] Katleho L Masita, Ali N Hasan, and Thokozani Shongwe. Deep learning in object detection: A review. In 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), pages 1–11. IEEE, 2020.
- [4] Ajeet Ram Pathak, Manjusha Pandey, and Siddharth Rautaray. Application of deep learning for object detection. *Procedia computer science*, 132:1706–1717, 2018