|  |  |
| --- | --- |
| Happy Monk | Veeresh kv |
| Machine Learning Assignment | July 2nd , 2021 |

# Definition

## Project Overview

Vision loss is a problem that affects people all around the world. As computers are getting better at understanding images due to advances in computer vision, the concept of a virtual assistant for the blind that could read text[[1]](#footnote-1), identify/spot objects2, or even describe a whole scene in natural language[[2]](#footnote-2) is becoming increasingly realistic.

## Problem Statement

The goal is to create a general text-reader running on Android smartphones; the tasks involved are the following:

1. Download and preprocess datasets
2. Train a classifier that can understand the data and predicts

The final application is expected to be useful for reading product labels, price tags, and other kinds of short, printed text.

## Metrics

Accuracy is a common metric for binary classifiers; it takes into account both true positives and true negatives with equal weight.

*accuracy* = *true* *positdivaetsa* *s*+*e* *tt**rsuizee* *negatives*

This metric was used when evaluating the classifier because false negatives and false positives both erode the user experience:

* False negatives result in either a longer delay between the user pointing the camera text and device speaking the text (“​**processing delay**​”) or in the worst case, completely prevent the application from reading said text

On the other hand, false positives make the application try to extract text from images that don’t contain any. This results in unnecessary computations on the remote server, which can be both costly and slow the application down. In the worst case, this might also result in the application reading gibberish.

## Algorithms and Techniques

The classifier is a ​Convolutional Neural Network,​ which is the state-of-the-art algorithm for most image processing tasks, including classification. It needs a large amount of training data compared to other approaches; fortunately, MNIST datasets are big enough. The algorithm outputs an assigned probability for each class; this can be used to reduce the number of false positives

using a ​**threshold**​. (The tradeoff is that this increases the number of false negatives.) The following parameters can be tuned to optimize the classifier:

* Classification ​**threshold**​ (see above)
* Training parameters
  + Training length (number of epochs)
  + Batch size (how many images to look at once during a single training step)
  + Solver type (what algorithm to use for learning)
  + Learning rate (how fast to learn; this can be dynamic)
  + Weight decay (prevents the model being dominated by a few “neurons”)
  + Momentum (takes the previous learning step into account when calculating the next one)
* Neural network architecture
  + Number of layers
  + Layer types (​convolutional​, ​fully-connected​, or ​pooling​)
  + Layer parameters (see links above)
* Preprocessing parameters (see the ​Data Preprocessing section)​

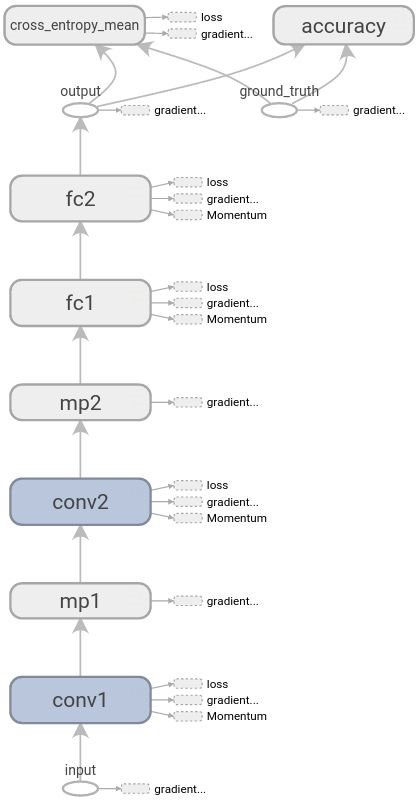
## Implementation

The implementation process can be split into two main stages:

1. The classifier training stage
2. The application development stage

During the first stage, the classifier was trained on the preprocessed training data. This was done in a Jupyter notebook (titled “Create and freeze graph”), and can be further divided into the following steps:

1. Load both the training and validation images into memory, preprocessing them as described in the previous section
2. Implement helper functions:
   1. get\_batch(...): Draws a random sample from the training/validation data
   2. fill\_feed\_dict(...): Creates a feed\_dict, which is a Python dictionary that contains all of the data required for a single training step (a batch of images, their labels, and the learning rate)
3. Define the network architecture and training parameters
4. Define the loss function, accuracy
5. Train the network, logging the validation/training loss and the validation accuracy
6. Plot the logged values
7. If the accuracy is not high enough, return to step 3
8. Save and freeze the trained network

**Fig. 5** The illustration on the right shows the computational graph, which includes the network architecture and also the variables that are only present during training. The acronyms can be read as follows: ❖ **FC**​: Fully connected layer

* **Conv**​: Convolutional layer
* **MP**​: Max pooling layer

The following can be seen by looking at the graph:

* The loss function is mainly composed of the mean cross entropy error.
* Large weights and biases of all four layers are penalized by using weight decay. (The weights are added to the loss function after they are multiplied by their specific weight decay rates.)

1. Jaderberg, Max et al. "Reading text in the wild with convolutional neural networks." ​*International Journal of Computer Vision*2 ​ 116.1 (2016): 1-20.

   "BlindTool" <​https://play.google.com/store/apps/details?id=the.blindtool&hl=en> (2016).​ [↑](#footnote-ref-1)
2. Devlin, Jacob et al. "Language models for image captioning: The quirks and what works." *arXiv*​ *preprint arXiv:1505.01809*4 ​ (2015).

   Lin, Tsung-Yi et al. "Microsoft coco: Common objects in context." ​*Computer Vision–ECCV 2014*​ (2014): 740-755. 5 Veit, Andreas et al. "COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images." *arXiv preprint arXiv:1601.07140*​ (2016).

   [↑](#footnote-ref-2)