Training Report

Deep Learning for Traffic Sign Recognition

Summer Training under guidance of

Mr. Amitava Das

Principal Scientist

Computational Instrumentation Department

CSIO-CSIR, Chandigarh

Submitted by

Amitojdeep Singh

BE (Hons.) Computer Science & Msc. (Hons.) Economics

Birla Institute of Technology & Sciences, Pilani

Acknowledgement

I would like to thank Mr. Amitava Das who has guided me throughout the duration of my training and the project work. He encouraged me to go into Deep Learning and work rigorously on the project. I am very grateful to my co-trainee Tushar Mehtani without whose constant support this project would never have been complete.

1. Introduction

Deep learning (also known as deep structured learning or hierarchical learning) is the application to learning tasks of artificial neural networks (ANNs) that contain more than one hidden layer. Deep learning is part of a broader family of machine learning methods based on learning data representations, as opposed to task specific algorithms. Learning can be supervised, partially supervised or unsupervised.

Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence.

In this project deep learning has been used to recognize traffic signs using German Traffic Sign Database. Accuracy of 99.38% has been achieved using the techniques discussed here in.

The German Traffic Sign Benchmark is a multi-class, single-image classification challenge held at the International Joint Conference on Neural Networks (IJCNN) 2011. We cordially invite researchers from relevant fields to participate: The competition is designed to allow for participation without special domain knowledge. Our benchmark has the following properties:

- Single-image, multi-class classification problem
- More than 40 classes
- More than 50,000 images in total
- Large, lifelike database

2. Motivation

Recognition of traffic signs is a challenging real-world problem of high industrial relevance. Although commercial systems have reached the market and several studies on this topic have been published, systematic unbiased comparisons of different approaches are missing and comprehensive benchmark datasets are not freely available.

Traffic sign recognition is a multi-class classication problem with unbalanced class frequencies. Traffic signs can provide a wide range of variations between classes in terms of color, shape, and the presence of pictograms or text. However, there exist subsets of classes (e. g., speed limit signs) that are very similar to each other.

The classier has to cope with large variations in visual appearances due to illumination changes, partial occlusions, rotations, weather conditions, etc.

Humans are capable of recognizing the large variety of existing road signs with close to 100% correctness. This does not only apply to real-world driving, which provides both context and multiple views of a single traffic sign, but also to the recognition from single images.

3. Task Description

The project task is a multi-class classification problem. Participating algorithms need to classify single images of traffic signs their performance will be determined based on the 0/1 loss function.

Although the problem domain of advanced driver assistance systems implies constraints on the runtime of employed algorithms, this project will not take processing times into account, as this puts too much emphasis on technical aspects like implementation issues and choice of programming language. These parameter are already very good while making predictions using deep learning.

The benchmark is designed in a way that allows scientists from different fields to contribute their results. As it excludes detection, tracking and temporal integration, prior knowledge in the domain is not required.

Example code concerning reading of images and writing results has already been provided ar GTSRB. In addition, there are many publicly available resources that can be used in order to

access state-of-the-art methods, e. g., the OpenCV library for computer vision and the Shark library for machine learning.

4. Dataset (GTSRB)

Overview

Single-image, multi-class classification problem
More than 40 classes
More than 50,000 images in total
Large, lifelike database
Reliable ground-truth data due to semi-automatic annotation
Physical traffic sign instances are unique within the dataset
(i.e., each real-world traffic sign only occurs once)

Structure

The training set archive is structures as follows:
One directory per class
Each directory contains one CSV file with annotations ("GT-<ClassID>.csv") and the training images
Training images are grouped by tracks
Each track contains 30 images of one single physical traffic sign

Image format

The images contain one traffic sign each
Images contain a border of 10 % around the actual traffic sign (at least 5 pixels) to allow for edge-based approaches
Images are stored in PPM format (Portable Pixmap, P6)

Image sizes vary between 15x15 to 250x250 pixels
Images are not necessarily squared
The actual traffic sign is not necessarily centered within the image.This is true for images that were close to the image border in the full camera image
The bounding box of the traffic sign is part of the annotations

Annotation format

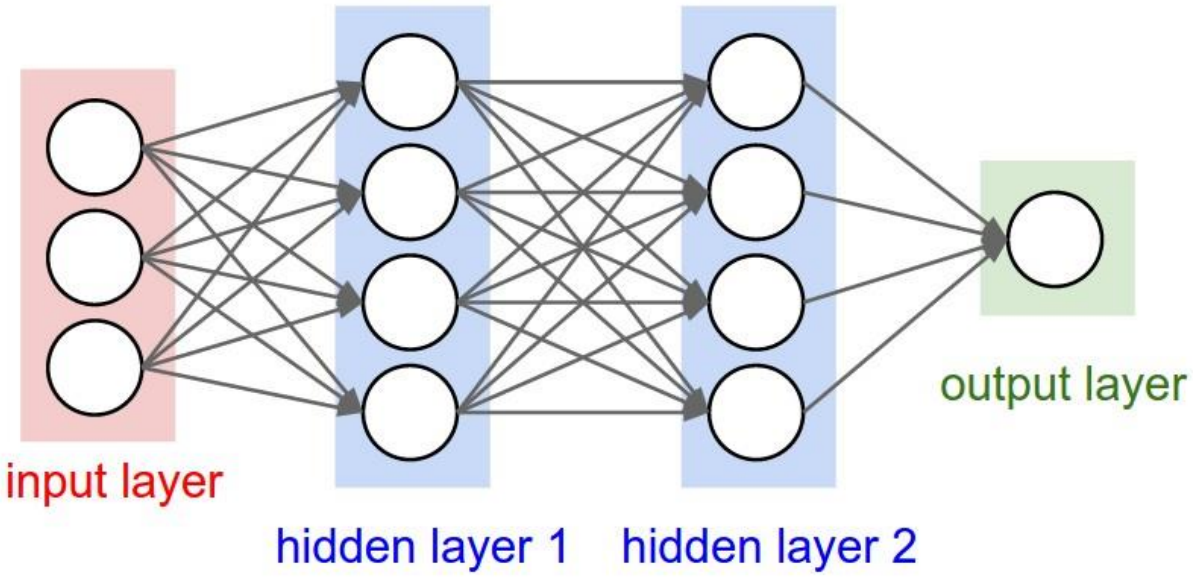
Annotations are provided in CSV files. Fields are separated by ";" (semicolon). Annotations contain the following information:

- Filename: Filename of corresponding image
- Width: Width of the image
- Height: Height of the image
- ROI.x1: X-coordinate of top-left corner of traffic sign bounding box
- ROI.y1: Y-coordinate of top-left corner of traffic sign bounding box
- ROI.x2: X-coordinate of bottom-right corner of traffic sign bounding box
- ROI.y2: Y-coordinate of bottom-right corner of traffic sign bounding box
  The training data annotations will additionally contain
  ClassId: Assigned class label

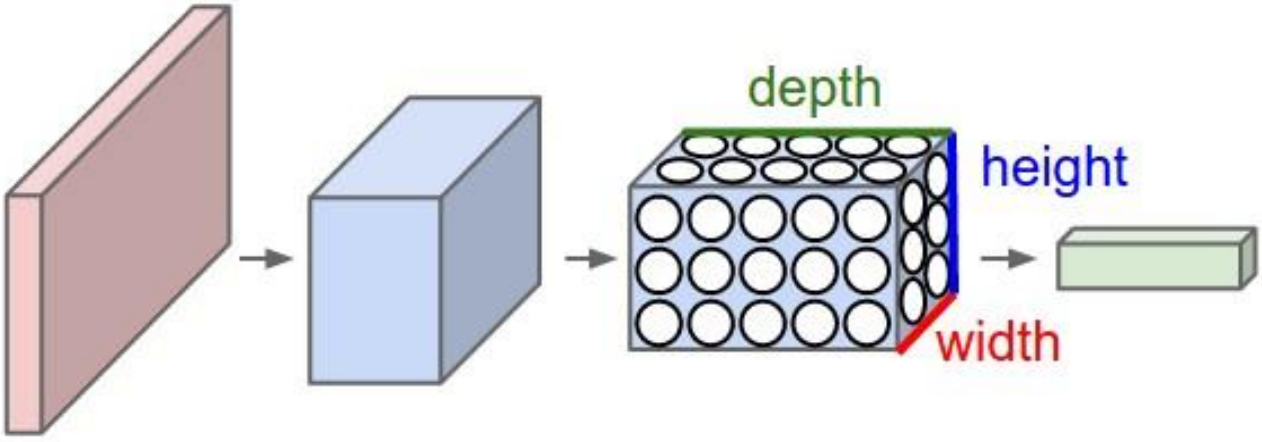
5. Deep Neural Networks & Related Terminology

Convolutional Neural Networks

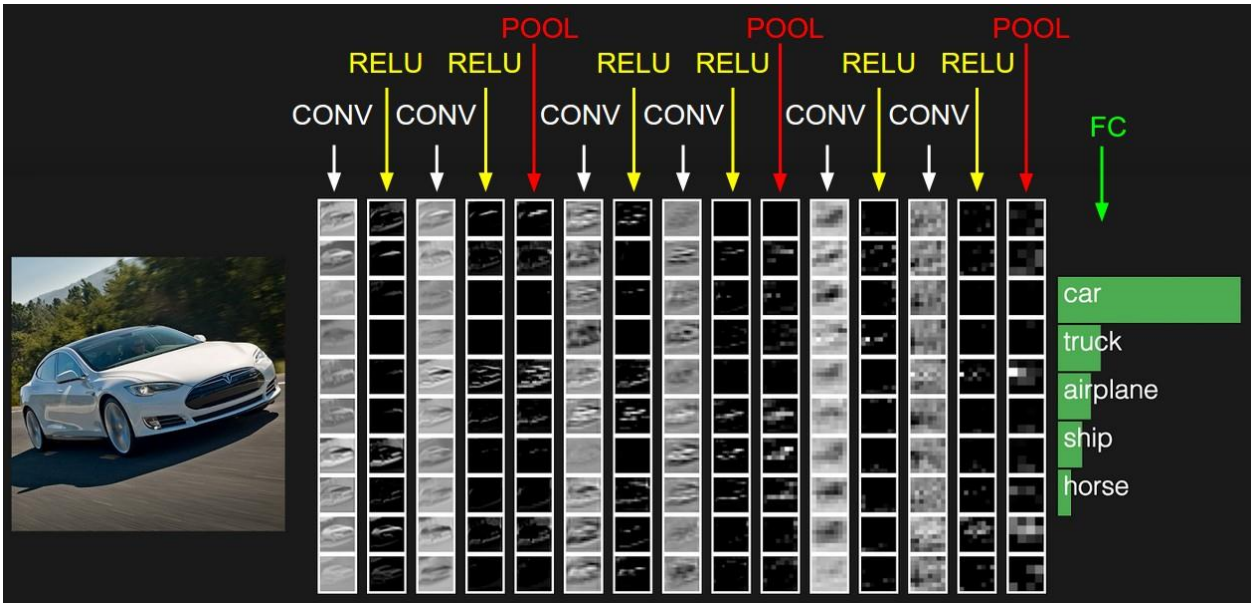
These are very similar to ordinary Neural Networks, they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other.



A regular 3-layer Neural Network



A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers.



The activations of an example ConvNet architecture.

DNN

It consists of a succession of convolutional and max-pooling layers, and each layer only receives connections from its previous layer. It is a general, hierarchical feature extractor that maps raw pixel intensities of the input image into a feature vector to be classified by several, usually 2 or 3, fully connected layers. All adjustable parameters are jointly optimized through minimization of the misclassification error over the training set.

Convolutional layer

Each convolutional layer performs a 2D convolution of its Mn−1 input maps with a filter of size K n x × K n y . The resulting activations of the Mn output maps are given by the sum of the Mn−1 convolutional responses which are passed through a nonlinear activation function: Y n j = f Mn−1 i=1 Y n−1 i ∗ Wn ij + b n j , (1) where n indicates the layer, Y is a map of size Mx × My, and Wij is a filter of size Kx × Ky connecting input map i with output map j, b n j is the bias of output map j, and ∗ is the valid 2D convolution. That is, for an input map Y n−1 of size Mn−1 x ×

Mn−1 y and a filter W of size K n x × K n y the output map Y n is of size Mn x = Mn−1 x − K n x + 1, Mn y = Mn−1 y −K n y +1. Note that the summation in Eq. (1) runs over all Mn−1 input maps.

Max-pooling layer

The biggest architectural difference between our DNN and the CNN of LeCun et al. (1998) is the use of max-pooling layers (Riesenhuber & Poggio, 1999; Scherer et al., 2010; Serre et al., 2005) instead of sub-sampling layers. The output of a max-pooling layer is given by the maximum activation over non-overlapping rectangular regions of size Kx × Ky. Max-pooling creates slight position invariance over larger local regions and down-samples the input image by a factor of Kx and Ky along each direction.

Classification layer

Kernel sizes of convolutional filters and max-pooling rectangles are chosen such that either the output maps of the last convolutional layer are down-sampled to 1 pixel per map, or a fully connected layer combines the outputs of the last convolutional layer into a 1D feature vector. The last layer is always a fully connected layer with one output unit per class in the recognition task. We use a softmax activation function for the last layer such that each neuron's output activation can be interpreted as the probability of a particular input image belonging to that class.

Training a single DNN

The training procedure of a single DNN is illustrated in Fig. 1b. A given dataset is preprocessed (P) before training starts, and then continually distorted (D) during training. Note that the preprocessing (details in Section 3.1) is not stochastic and is done for the whole dataset prior to training. Distortions on the other hand are stochastic and applied to each preprocessed image

Keras

A high-level neural networks API, written in Python and capable of running on top of either TensorFlow, CNTK or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

The Sequential model

It is a linear stack of layers. An example of sequential keras model.

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential([
    Dense(32, input_shape=(784,)),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

6. Model for Traffic Sign Prediction

The model used for traffic sign recognition in this project has the following basic structure:

```
def get_model_bn():
    model = Sequential([
        Lambda(get_x,input_shape=(3,50,50)),
        Convolution2D(32,3,3, activation='relu'),
        BatchNormalization(axis=1),
        Convolution2D(32,3,3, activation='relu'),
        MaxPooling2D(),
        BatchNormalization(axis=1),
        Convolution2D(64,3,3, activation='relu'),
        BatchNormalization(axis=1),
        Convolution2D(64,3,3, activation='relu'),
        MaxPooling2D(),
        Convolution2D(128,3,3, activation='relu'),
        BatchNormalization(axis=1),
        Convolution2D(128,3,3, activation='relu'),
        BatchNormalization(axis=1),
        MaxPooling2D(),
        Flatten(),
        BatchNormalization(),
        Dense(4096, activation='relu'),
        Dropout(0.5),
```

```
        Dense(1024, activation='relu'),
        Dropout(0.5),
        BatchNormalization(),
        Dense(43, activation='softmax')
        ])
    model.compile(Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])
    return model
```

It has 6 convolutional layers and 3 dense layers. Dropout of 0.5 has been used for dense layers. Overall the architecture is a simpler version of VGG 16 model. Adam optimizer has been used to control the learning rate. Loss has been computed as categorical cross entropy as there are 43 categories of traffic signs here.

Target image size was chosen as 50*50 as it ensured clear enough images and was in accordance with prior work in the area. Training - validation split of 0.15 was used. Model was trained for 20 epochs as the training accuracy stabilized and validation accuracy became quite high.

Following is an example of the process of training.

```
model.fit_generator(
        train_generator,
        samples_per_epoch=train_generator.n,
        nb_epoch=1,
        validation_data=validation_generator,
        nb_val_samples=validation_generator.n)
Epoch 1/1
35288/35288 [==============================] - 463s - loss: 0.9445 -
acc: 0.7276 - val_loss: 0.1722 - val_acc: 0.9464
Epoch 1/1
35288/35288 [==============================] - 504s - loss: 0.2246 -
acc: 0.9311 - val_loss: 0.0296 - val_acc: 0.9913
Epoch 1/1
35288/35288 [==============================] - 510s - loss: 0.1317 -
acc: 0.9593 - val_loss: 0.0398 - val_acc: 0.9880
Epoch 1/1
35288/35288 [==============================] - 517s - loss: 0.1176 -
acc: 0.9646 - val_loss: 0.1788 - val_acc: 0.9436
Epoch 1/1
35288/35288 [==============================] - 515s - loss: 0.0962 -
acc: 0.9715 - val_loss: 0.0086 - val_acc: 0.9972
Epoch 1/1
```

```
35288/35288 [==============================] - 540s - loss: 0.0816 -
acc: 0.9751 - val_loss: 0.0315 - val_acc: 0.9929
Epoch 1/1
35288/35288 [==============================] - 518s - loss: 0.0797 -
acc: 0.9766 - val_loss: 0.0193 - val_acc: 0.9957
Epoch 1/1
35288/35288 [==============================] - 521s - loss: 0.0724 -
acc: 0.9781 - val_loss: 0.0068 - val_acc: 0.9990
Epoch 1/1
35288/35288 [==============================] - 524s - loss: 0.0641 -
acc: 0.9800 - val_loss: 0.0045 - val_acc: 0.9985
Epoch 1/1
35288/35288 [==============================] - 521s - loss: 0.0525 -
acc: 0.9849 - val_loss: 0.0021 - val_acc: 0.9992
Epoch 1/1
35288/35288 [==============================] - 524s - loss: 0.0554 -
acc: 0.9834 - val_loss: 0.0069 - val_acc: 0.9977
Epoch 1/1
35288/35288 [==============================] - 520s - loss: 0.0531 -
acc: 0.9845 - val_loss: 0.0051 - val_acc: 0.9990
Epoch 1/1
35288/35288 [==============================] - 522s - loss: 0.0547 -
acc: 0.9848 - val_loss: 0.0047 - val_acc: 0.9990
Epoch 1/1
35288/35288 [==============================] - 520s - loss: 0.0415 -
acc: 0.9882 - val_loss: 0.0054 - val_acc: 0.9985
Epoch 1/1
35288/35288 [==============================] - 519s - loss: 0.0397 -
acc: 0.9878 - val_loss: 0.0117 - val_acc: 0.9972
Epoch 1/1
35288/35288 [==============================] - 522s - loss: 0.0651 -
acc: 0.9828 - val_loss: 0.0053 - val_acc: 0.9990
Epoch 1/1
35288/35288 [==============================] - 522s - loss: 0.0400 -
acc: 0.9885 - val_loss: 0.0026 - val_acc: 0.9995
Epoch 1/1
35288/35288 [==============================] - 526s - loss: 0.0289 -
acc: 0.9918 - val_loss: 0.0039 - val_acc: 0.9992
Epoch 1/1
35288/35288 [==============================] - 526s - loss: 0.0319 -
acc: 0.9909 - val_loss: 0.0032 - val_acc: 0.9992
Epoch 1/1
35288/35288 [==============================] - 520s - loss: 0.0375 -
acc: 0.9883 - val_loss: 0.0017 - val_acc: 0.9992
```

7. Results

The model was used to make predictions on the test dataset and a maximum accuracy of 99.38% which is higher than any human performance based approach. The result is shown below:

TEAM	METHOD	TOTAL	SUBSET (All signs)	DETAILS
[140] lizichen	random test	99.99%	99.99%	
[156] aag-us	CNN ST3 exp2 e21	99.71%	99.71%	
[150] aag-us	CNN STs exp2	99.68%	99.68%	
[118] cacophonouscockatoos	ensemble3	99.58%	99.58%	
[19] wgy@HIT501	2-stage HOG+SVM	99.52%	99.52%	
[3] IDSIA	Committee of CNNs	99.46%	99.46%	
[163] CSIO_Trainees	ensemble8	99.38%	99.38%	Details
[164] CSIO_Trainees	ensemble13	99.37%	99.37%	Details
[161] CSIO_Trainees	CNN ensembled	99.26%	99.26%	Details
[8] INI-RTCV	Human (best individual)	99.22%	99.22%	
[165] CSIO_Trainees	ensemble10	99.15%	99.15%	Details
[155] COSFIRE	Color-blob-based COSFIRE filters for object recogn	98.97%	98.97%	
[1] INI-RTCV	Human Performance	98.84%	98.84%	
[17] lcad-ufes	VGRAM WNN RGB Final 2	98.73%	98.73%	
[162] CSIO_Trainees	CNN ensembled + data augmentation	98.71%	98.71%	Details
[18] lcad-ufes	VGRAM WNN RGB Final 3	98.69%	98.69%	
[22] lcad-ufes	VG-RAM WNN M 50	98.69%	98.69%	
[15] lcad-ufes	VG-RAM WNN RGB Final	98.65%	98.65%	
[154] matthew-kleinsmith	VGG-style, ensemble, dropout	98.65%	98.65%	
[160] CSIO_Trainees	CNN with data augmentation	98.60%	98.60%	Details

8. References

1. CireşAn, Dan, et al. "Multi-column deep neural network for traffic sign classification." *Neural Networks* 32 (2012): 333-338.
2. www.courses.fast.ai
3. https://en.wikipedia.org/wiki/Deep_Learning