Traffic Sign Recognition

Ioannis Ioannidis
Dimitrios Patiniotis Spyropoulos

University of Piraeus
NCSR Demokritos
Athens

27-06-2022

Outline

- Introduction
- Dataset Description
- Theoretical Background
- Experiments
- Conclusions

Introduction

Motivation

- Road safety is attracting the attention of many researchers around the world.
- Systems for the detection, classification, and recognition of road signs have become a very important branch on this domain.
- Traffic sign detectors effectively assist drivers in the process of driving and keep them driving more safely as they inform them about current road situations and potential hazards.
- Constitute the "eyes" of self-driving cars.

Dataset Description

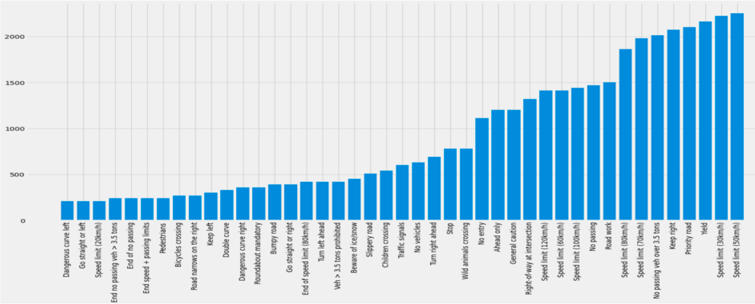
Data Source

- Dataset for "German Traffic Sign Recognition Benchmark", a multi-category classification competition held at IJCNN 2011.
- Comprehensive, lifelike dataset of more than 40,000 traffic sign images.
- Reflects the strong variations in visual appearance of signs due to distance, illumination, weather conditions, partial occlusions, and rotations.

Dataset Description

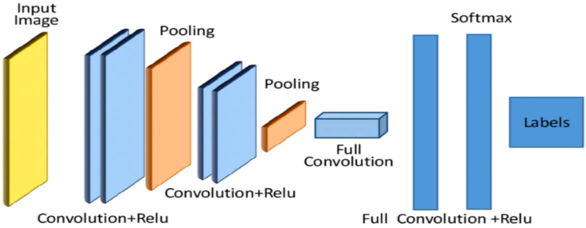
Classes Distribution

- Dataset contains 30.209 training and 12.630 test images.
- It comprises 43 classes with unbalanced class frequencies.
- This is quite logical since some signs like “Keep speed below 30 K-mph” or “A bump ahead” appears more often then signs like “Road under construction ahead”



CNN

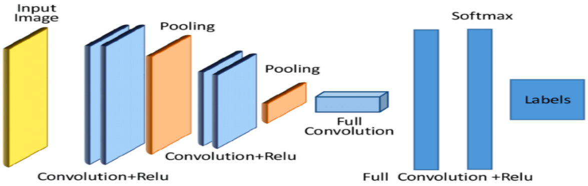
- Convolutional Neural Networks (CNNs) is a state of the art pattern recognition method in computer vision.
- Unlike traditional neural networks, which work with one-dimensional feature vectors, a CNN takes a two-dimensional image and consequentially processes it with convolutional layers.



A generic CNN Architecture.

CNN

- A CNN is composed of input and output layers and multiple hidden layers, which can be divided into a convolution layer, a pooling layer, a rectified linear unit layer, and a fully connected layer.
- Each convolutional layer consists of a set of trainable filters and computes dot productions between these filters and layer input to obtain an activation map.
- These filters are also known as kernels and allow detecting the same features in different locations.



A generic CNN Architecture.

Experiments

Description

- Experiment I: Optimizers Comparison
- Experiment II: Training Size Modification
- Experiment III: Number of Convolutional Layers

Experiment I

Optimizers Comparison

- **Stohastic Gradient Decent**

Minimizes an objective function $J(\theta)$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ w.r.t. to the parameters.

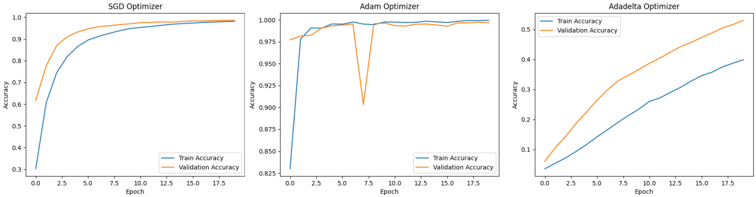
- **Adam Optimizer**

Accelerates the gradient descent algorithm by taking into consideration the "exponentially weighted average" of the gradients. Using averages makes the algorithm converge towards the minima in a faster pace.

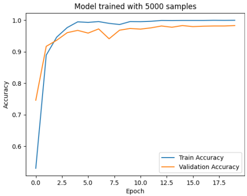
- **AdaDelta Optimizer**

Dynamically adapts over time using only first order information and has minimal computational overhead. It requires no manual tuning of a learning rate and appears robust to noisy gradient information, different model architecture choices, various data modalities and selection of hyperparameters.

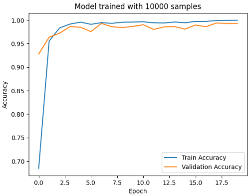
Optimizers Comparison Results



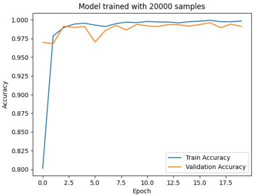
Training Size Modification



(a) 5s/epoch



(b) 9s/epoch



(c) 20s/epoch

Reducing Convolution Layers

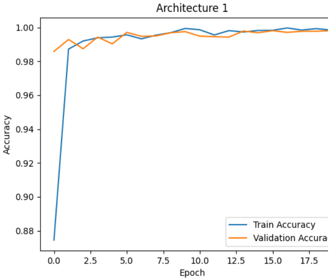
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 16)	448
conv2d_1 (Conv2D)	(None, 26, 26, 32)	4640
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
batch_normalization (BatchNormalization)	(None, 13, 13, 32)	128
conv2d_2 (Conv2D)	(None, 11, 11, 64)	18496
conv2d_3 (Conv2D)	(None, 9, 9, 128)	73856
max_pooling2d_1 (MaxPooling2D)	(None, 4, 4, 128)	0
batch_normalization_1 (BatchNormalization)	(None, 4, 4, 128)	512
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 512)	1049088
batch_normalization_2 (BatchNormalization)	(None, 512)	2048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 43)	22059

Total params: 1,171,275
Trainable params: 1,169,931
Non-trainable params: 1,344

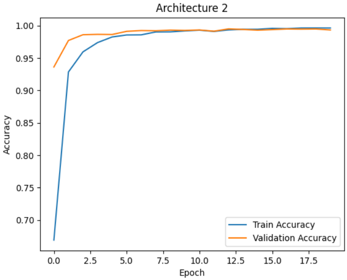
Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 28, 28, 16)	448
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 16)	0
batch_normalization_3 (BatchNormalization)	(None, 14, 14, 16)	64
conv2d_5 (Conv2D)	(None, 12, 12, 32)	4640
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 32)	0
batch_normalization_4 (BatchNormalization)	(None, 6, 6, 32)	128
flatten_1 (Flatten)	(None, 1152)	0
dense_2 (Dense)	(None, 128)	147584
batch_normalization_5 (BatchNormalization)	(None, 128)	512
dropout_1 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 43)	5547

Total params: 158,923
Trainable params: 158,571
Non-trainable params: 352

Reducing Convolution Layers Results



(a) Baseline CNN



(b) Reduced Architecture

Conclusion

- Despite the gradual increase of SGD, Adam seems to perform better and faster, while AdaDelta is way too slow.
- Conserning training size, we observed that while accuracy seems not to be that much affected, training time is significantly reduced when we decrease the train sample.
- Finally, reducing the model's convolutions and parameters, we achieve similar performance, with lower computational cost and time.