Traffic Sign Classifier

An approach to solve German Traffic Classification Problem by using CNN

Manuel Scurti - Mingjie Ye

Problem

The german traffic sign benchmark (GTSRB) is a multi-class classification challenge. The benchmark has the following properties:

- Multi-class classification problem
- More than 40 classes
- More than 50000 images in total

Image preprocessing

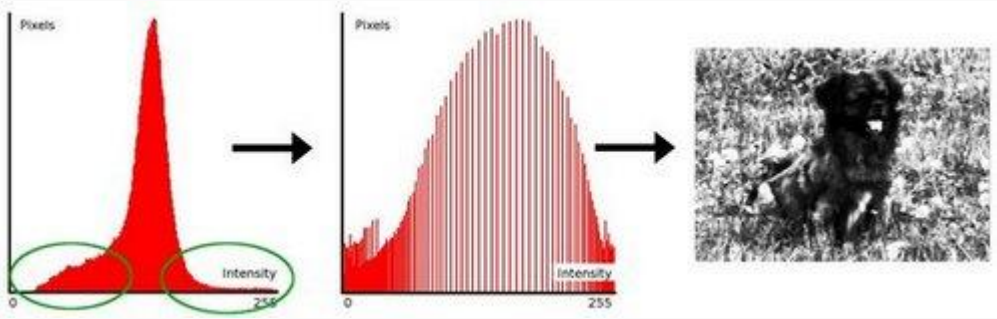
Preprocessing of images consisted in:

1. Histogram equalization to improve the contrast of the traffic signs images
2. Scaling from 224x224 to 48x48 to speed up the learning process, little image input but deeper convolutional layers

After histogram equalization

Image preprocessing - Histogram Eq

- An image histogram is a graphical representation of the intensity distribution of an image. It quantifies the number of pixels for each intensity value considered.
- Histogram equalization is a method that improves the contrast in an image, in order to stretch out the intensity range. How?



Source: OpenCV

Network Architecture (1 / 3)

General overview of the network

- 6 convolutional layers and 1 dense layer after flattening
- Activation function: Leaky ReLU
- Regularization method: Batch Normalization, Max Pooling
- Optimizer: SGD + Nesterov with learning rate of 0.01
- Training epochs: 60 with data augmentation, 30 without
- Total number of parameters: 2,672,587

Network Architecture (2 / 3)

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 48, 48, 32)	896
activation_1 (Activation)	(None, 48, 48, 32)	0
conv2d_2 (Conv2D)	(None, 48, 48, 32)	9248
batch_normalization_1 (Batch	(None, 48, 48, 32)	128
activation_2 (Activation)	(None, 48, 48, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 24, 24, 32)	0
conv2d_3 (Conv2D)	(None, 24, 24, 64)	18496
batch_normalization_2 (Batch	(None, 24, 24, 64)	256
activation_3 (Activation)	(None, 24, 24, 64)	0
conv2d_4 (Conv2D)	(None, 24, 24, 64)	36928
batch_normalization_3 (Batch	(None, 24, 24, 64)	256
activation_4 (Activation)	(None, 24, 24, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 64)	0
conv2d_5 (Conv2D)	(None, 12, 12, 128)	73856
batch_normalization_4 (Batch	(None, 12, 12, 128)	512
activation_5 (Activation)	(None, 12, 12, 128)	0
conv2d_6 (Conv2D)	(None, 12, 12, 128)	147584
batch_normalization_5 (Batch	(None, 12, 12, 128)	512
activation_6 (Activation)	(None, 12, 12, 128)	0
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 128)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_1 (Dense)	(None, 512)	2359808
batch_normalization_6 (Batch	(None, 512)	2048
activation_7 (Activation)	(None, 512)	0
dense_2 (Dense)	(None, 43)	22059
activation_8 (Activation)	(None, 43)	0

Network Architecture (3 / 3)

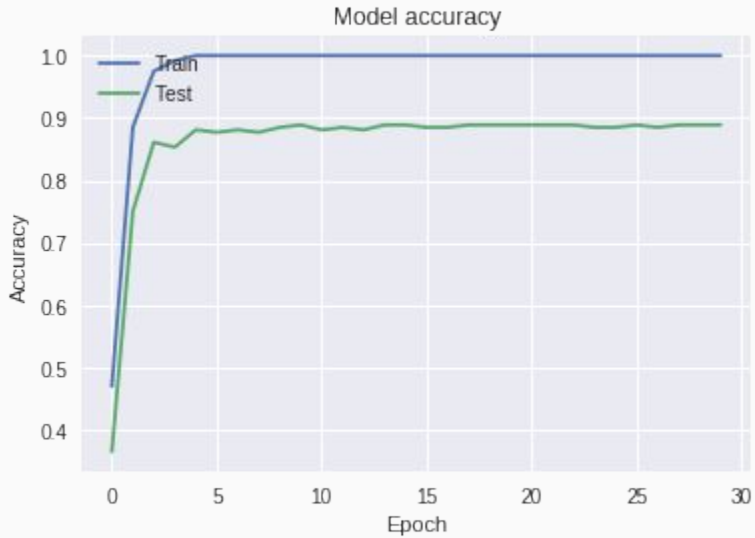
The convolutional layers have been configured with these parameters:

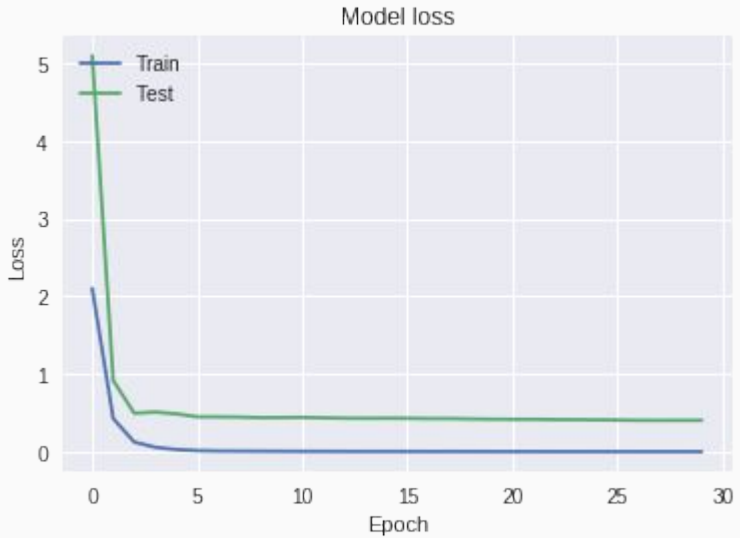
- First two convolutional layers: 3x3x32
- Seconds two convolutional layers: 3x3x64
- Last two convolutional layers: 3x3x128
- Pooling was configured with a grid of 2x2 using the max function

  All with stride 1 and padding “same” to have the same volume size between convolutional layers

  We also tried different kernel sizes: 5x5, 7x7

Performance





Time: 0.1309s Test Accuracy: 0.9391

Improving with Data Augmentation

- width_shift_range=0.2
- height_shift_range=0.2
- shear_range=0.2
- zoom_range=0.2
- fill_mode='nearest'

Advantages: Improved generalization of the network

Disadvantages: Oscillating convergence

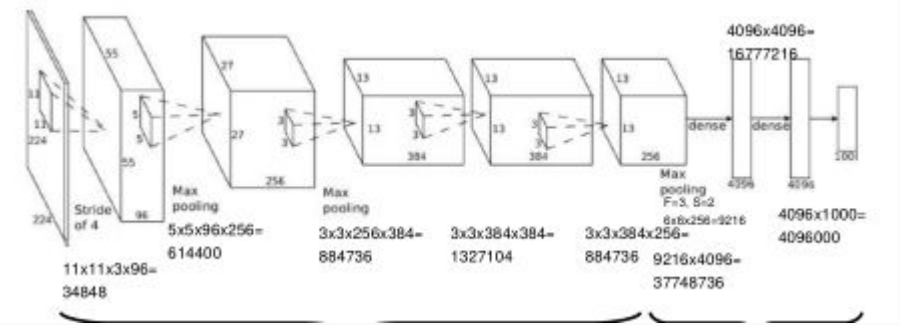
(trembling accuracy and loss)

Performances with Data Augmentation

Time: 0.1735s Time increase because we trained the augmented data for more epochs
Test Accuracy: 0.9584 [IMPROVED!]

Process followed

1. Architectural idea
   a. Pyramid model
2. Regularization techniques
   a. BatchNormalization
3. Better preprocessing of the images
4. Scaling of the images to speed up
5. Data augmentation
6. Tuning of epochs, learning rates and other hyperparameters



Bibliography

- OpenCV documentation - https://docs.opencv.org/
- Equalizing colored images - https://stackoverflow.com/questions/31998428/opencv-python-equalizehist-colored-image
- Vision por Computador - Image processing chapter - UPM course
- Keras documentation - https://keras.io/layers/convolutional/

Thank you!

Questions?