Traffic Sign Classification

Lewis Collum

Updated: February 14, 2020

Started 02/09/2020

Dataset: German Traffic Sign Recognition Benchmark (GTSRB)

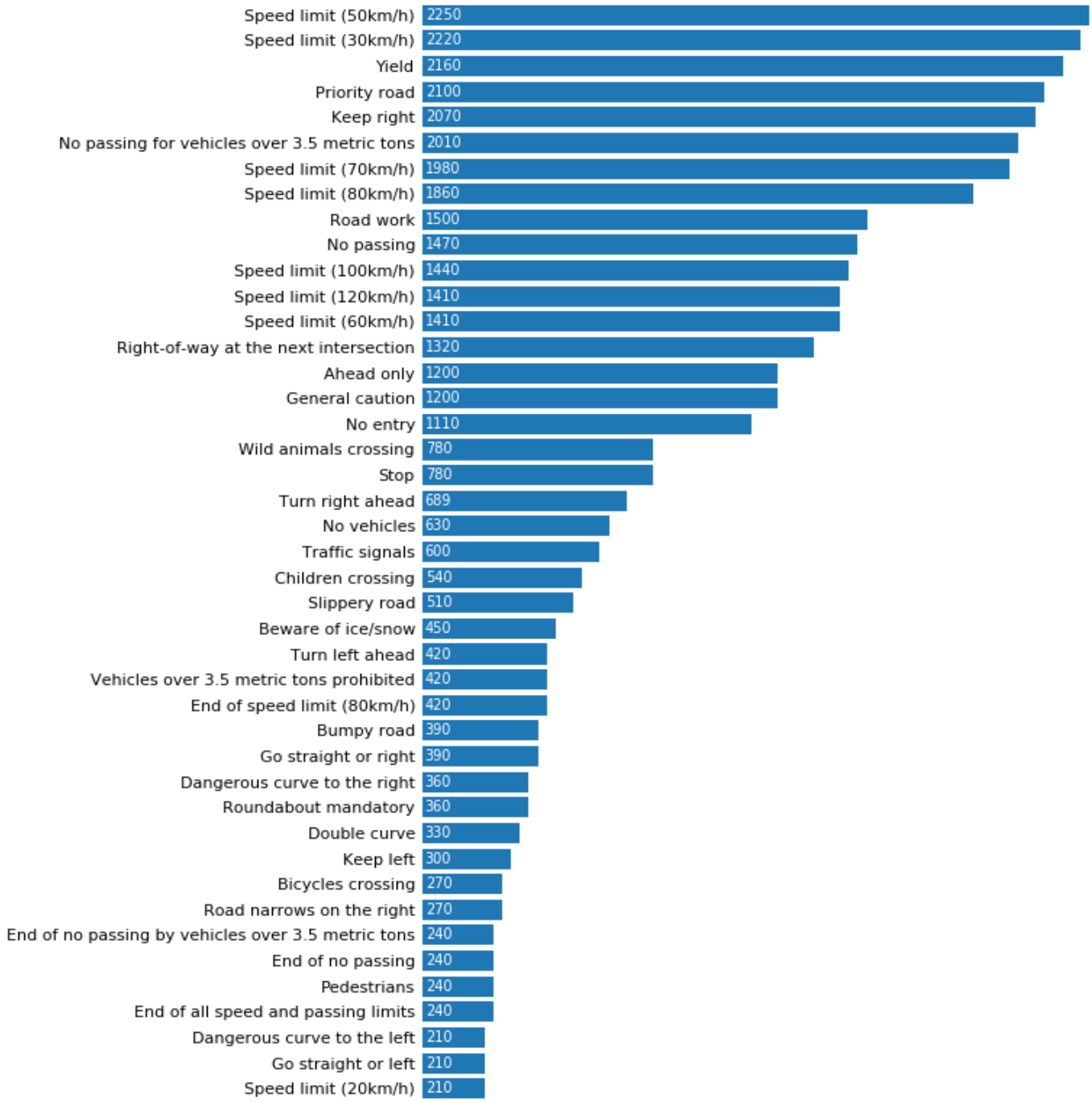
Downloading the GTSRB

The following downloads the dataset, unzips it, and moves the internal directories around (and changes their name) to match my prefered directory convention.

```
trainSet=GTSRB_Final_Training_Images.zip
testSet=GTSRB_Final_Test_Images.zip
signNames=signnames.csv
dataDirectory="data"

function main() {
    downloadGtsrbTrainSet
    dowloadGtsrbTestSet
    downloadSignNames
    configureDataDirectory
    echo "DONE!"
}

function downloadGtsrbTrainSet() {
    if [[ ! -d "$dataDirectory/train" ]]; then
        wget https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/$trainSet
        unzip $trainSet -d "$dataDirectory"
        rm -f $trainSet
    fi;
}

function downloadGtsrbTrainSet() {
    if [[ ! -d "$dataDirectory/test" ]]; then
        wget https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/$testSet
        unzip $testSet -d "$dataDirectory"
        rm -f $testSet
    fi;
}

function downloadSignNames() {
    wget https://raw.githubusercontent.com/udacity/CarND-Traffic-Sign-Classifier-Project/master/"$sign
}
```

```
function configureDataDirectory() {
    mv "$signNames" "$dataDirectory/names.csv"

    [[ -d "data/GTSRB/Final_Training/Images" ]] &&
        mv "data/GTSRB/Final_Training/Images" "data/train"

    [[ -d "data/GTSRB/Final_Test/Images" ]] &&
        mv "data/GTSRB/Final_Test/Images" "data/test"

    rm -rf "data/GTSRB"
}

main
```

Distribution of Classes



Path Organization

Relies on the user running all code from the `source` directory

```
import os
from datetime import datetime
```

```
directory = "../runs"
current = os.path.join(directory, ".current")

class Run:
    def __init__(self, runName):
        run = os.path.join(directory, runName)
        self.model = os.path.join(run, "model.h5")
        self.log = os.path.join(run, "log.csv")
        self.accuracy = os.path.join(run, "accuracy.png")
        self.modelDiagram = os.path.join(run, "model.png")
        self.modelSummary = os.path.join(run, "modelSummary")

def make():
    runName = f"{datetime.now():%m-%d_%H%M}"
    newRun = os.path.join(directory, runName)
    os.mkdir(newRun)
    with open(current, 'w') as f:
        f.write(runName)

def loadFromName(runName):
    return Run(runName)

def loadCurrent():
    with open(current) as f:
        return loadFromName(f.readline())

def has(path):
    return os.path.isfile(path)
```

```
import os

names = "../data/names.csv"
train = "../data/train"
```

Batch: Preperation with ImageDataGenerator and DirectoryIterator

batch.py

```
from keras.preprocessing.image import ImageDataGenerator
import numpy
import matplotlib.pyplot as pyplot
from textwrap import wrap
import pandas

import preprocessing
import path

signNames = pandas.read_csv(path.data.names)['SignName'].values
imageSize = 32
size = 32
```

```
batchGenerator = ImageDataGenerator(
    rescale = 1.0/255,
    preprocessing_function = preprocessing.execute,
    validation_split = 0.2)

trainIterator = batchGenerator.flow_from_directory(
    directory = path.data.train,
    batch_size = size,
    shuffle = True,
    target_size = (imageSize, imageSize),
    color_mode = 'grayscale',
    subset = 'training')

validationIterator = batchGenerator.flow_from_directory(
    directory = path.data.train,
    batch_size = size,
    shuffle = True,
    target_size = (imageSize, imageSize),
    color_mode = 'grayscale',
    subset = 'validation')

_images, _labels = trainIterator.next()
classCount = len(_labels[0])
sampleClasses = trainIterator.labels
sampleSize = trainIterator.n
imageShape = _images[0].shape

def classFromLabelsAt(labels, index):
    return numpy.where(labels[index] == 1)[0][0]

def signNameFromLabelsAt(labels, index):
    return signNames[classFromLabelsAt(labels, index)]

def plot(images, labels, columns=5, rows=5):
    figure, axes = pyplot.subplots(rows, columns, figsize=(8,2*rows))
    figure.subplots_adjust(hspace = .6)

    for n in range(min(columns*rows, len(images))):
        if len(images[n, 0, 0]) == 1:
            figure.axes[n].imshow(images[n].reshape((imageSize, imageSize)), cmap='gray')
        else:
            figure.axes[n].imshow(images[n])

        title = signNameFromLabelsAt(labels, n).title()
        wrappedTitle = "\n".join(wrap(title, 18))
        figure.axes[n].set_title(wrappedTitle, fontsize=10)

    for subplotAxes in figure.axes: subplotAxes.axis('off')
    figure.tight_layout()
```

Preprocessing

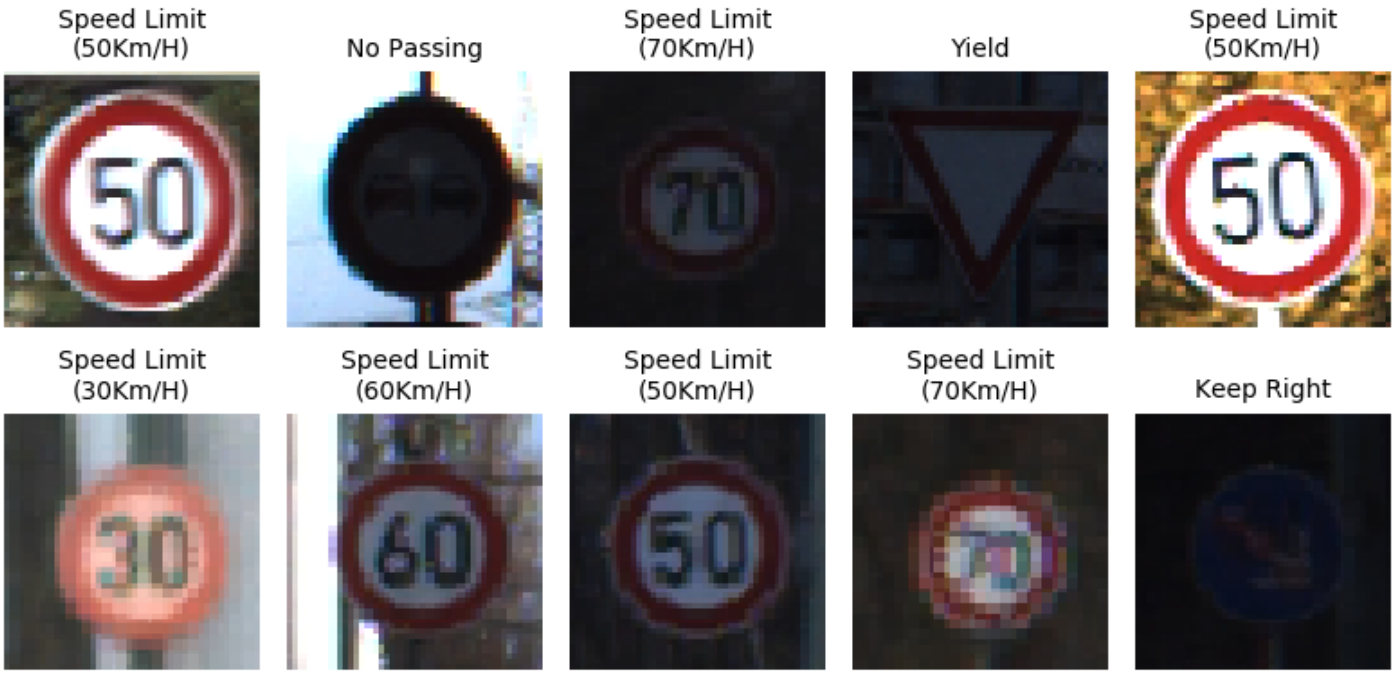
keras runs the `preprocessing_function` before scaling. So, a pixel that is 255.0, then becomes 255 (right before clahe is applied), then 255.0 at the end; finally, keras scales it to 1.0 for training. `preprocessing.py`

```
import cv2
import numpy

clahe = cv2.createCLAHE(tileGridSize=(2,2), clipLimit=15.0)

def execute(image):
    resultImage = clahe.apply(image.astype(numpy.uint8))
    return resultImage.reshape(image.shape[0], image.shape[1], 1).astype(numpy.float32)
```

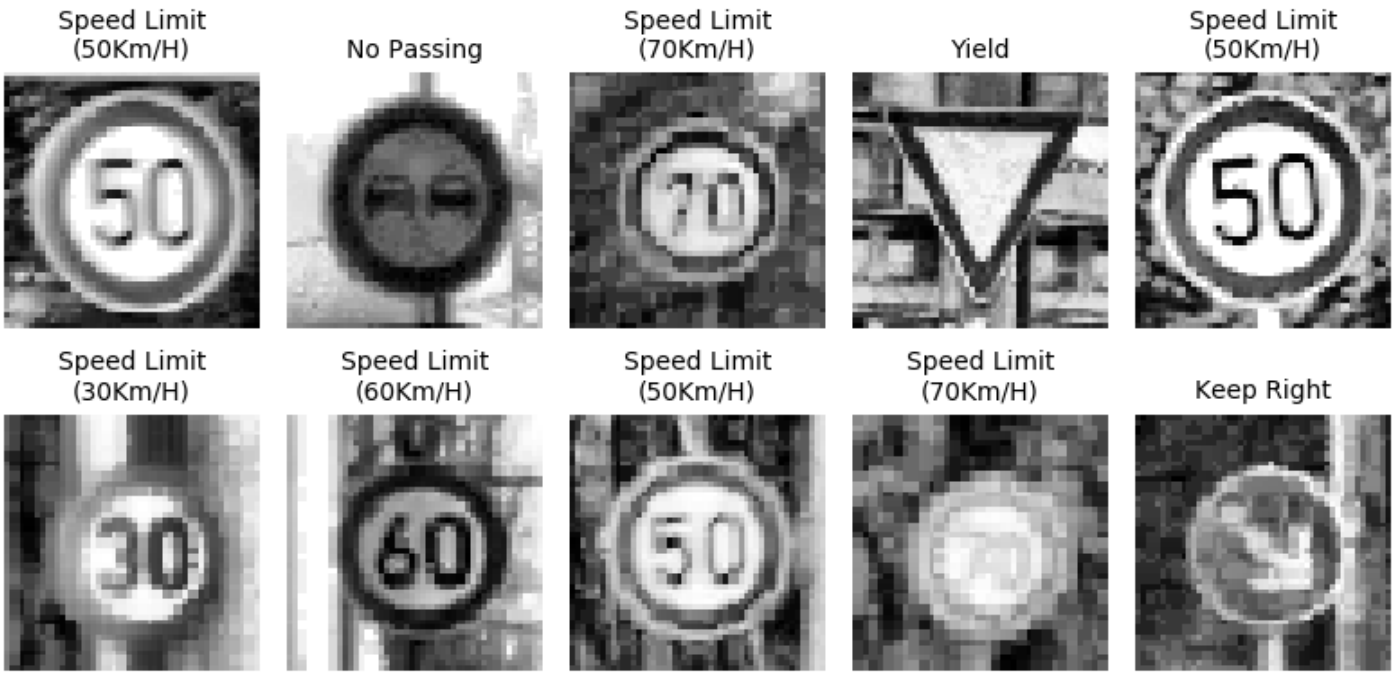
Batch with Histogram Equalization (Contrast Balancing)

```
import keras
import matplotlib.pyplot as pyplot
import numpy
import cv2
import PIL

import path
import preprocessing
import batch

generator = keras.preprocessing.image.ImageDataGenerator()
iterator = generator.flow_from_directory(path.data.train, batch_size=10, shuffle=True, target_size=(ba

images, labels = next(iterator)
claheImages = numpy.array([preprocessing.execute(cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)) for image in
batch.plot(images/255, labels, columns=5, rows=2)
#pyplot.savefig('../figure/batchClaheComparison_noClahe.png')
batch.plot(claheImages, labels, columns=5, rows=2)
pyplot.show()
#pyplot.savefig('../figure/batchClaheComparison_clahe.png')
```

Before



After



Model: LeNet

`model.py`

```
import keras
import keras.layers as layers
```

```python
import batch

model = keras.models.Sequential()

for i in range(3):
    model.add(layers.Conv2D(filters=32*2**i, kernel_size=(3, 3), activation='relu', input_shape=batch.
    model.add(layers.Dropout(0.1))
    model.add(layers.MaxPool2D(pool_size=(2, 2)))

model.add(layers.Flatten())

model.add(layers.Dense(units=120, activation='relu'))
model.add(layers.Dense(units=84, activation='relu'))
model.add(layers.Dense(units=batch.classCount, activation = 'softmax'))

if __name__ == '__main__':
    print(model.summary())
```

```
Found 31368 images belonging to 43 classes.
Found 7841 images belonging to 43 classes.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 30, 30, 32)        320
_________________________________________________________________
dropout_1 (Dropout)          (None, 30, 30, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
dropout_2 (Dropout)          (None, 13, 13, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 128)         73856
_________________________________________________________________
dropout_3 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 120)               61560
_________________________________________________________________
dropout_4 (Dropout)          (None, 120)               0
_________________________________________________________________
dense_2 (Dense)              (None, 84)                10164
_________________________________________________________________
dropout_5 (Dropout)          (None, 84)                0
_________________________________________________________________
```

```
dense_3 (Dense)              (None, 43)                3655
_________________________________________________________________
dropout_6 (Dropout)          (None, 43)                0
=================================================================
Total params: 168,051
Trainable params: 168,051
Non-trainable params: 0
_________________________________________________________________
None
```

Training

training.py

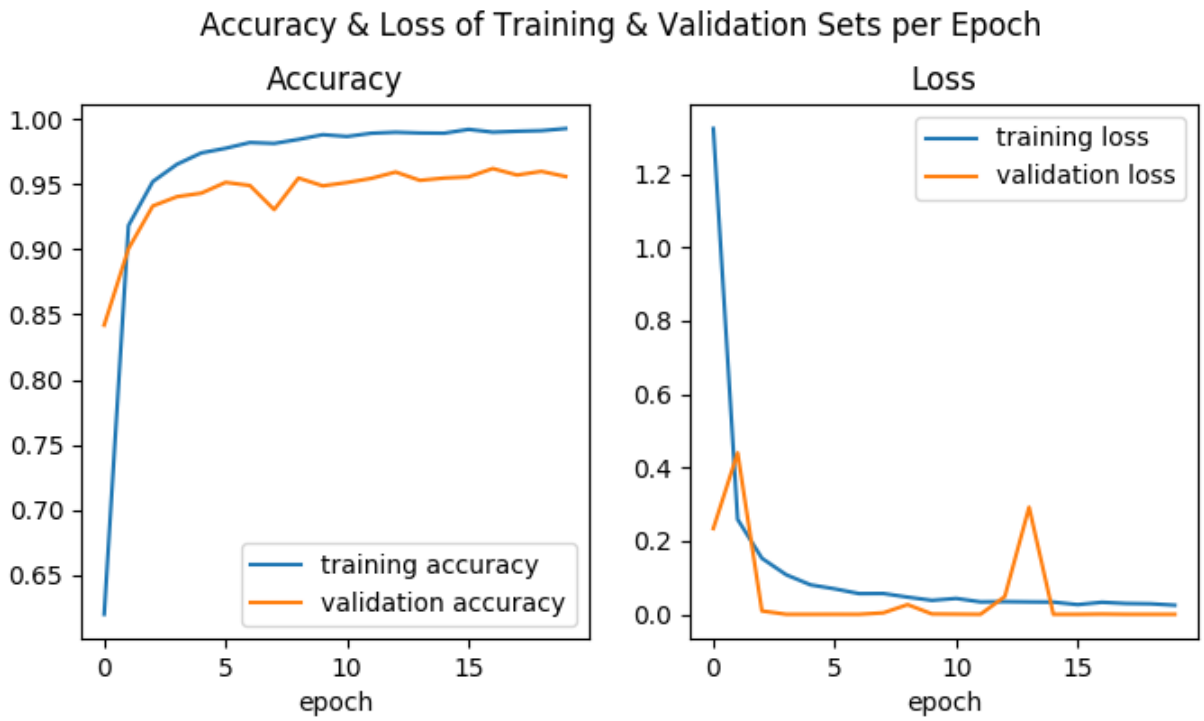
```
import keras
from keras.callbacks import CSVLogger

import path
from model import model
import batch
import logger

path.run.make()
run = path.run.loadCurrent()

model.compile(
    loss = 'categorical_crossentropy',
    optimizer = keras.optimizers.Adam(),
    metrics = ['accuracy'])

model.fit_generator(
    batch.trainIterator,
    validation_data = batch.validationIterator,
    steps_per_epoch = batch.trainIterator.n/batch.size,
    epochs = 8,
    callbacks = [
        keras.callbacks.CSVLogger(run.log, separator=',', append=False)
    ])

model.save(run.model)
logger.addModelDiagram(run)
logger.addModelSummary(run)
logger.addAccuracyPlot(run)
```

Logging Training Runs

```
import pandas
import matplotlib.pyplot as pyplot
import keras
import path
import os
```

```
def addAccuracyPlot(run):
    log = pandas.read_csv(run.log)

    figure, axes = pyplot.subplots(1, 2, figsize=(8, 4))
    axes[0].set_title('Accuracy')
    axes[0].plot(log['epoch'], log['accuracy'], log['val_accuracy'])
    axes[0].legend(['training accuracy', 'validation accuracy'])
    axes[0].set_xlabel('epoch')

    axes[1].set_title('Loss')
    axes[1].plot(log['epoch'], log['loss'], log['val_loss'])
    axes[1].legend(['training loss', 'validation loss'])
    axes[1].set_xlabel('epoch')

    figure.subplots_adjust(top=0.85)
    figure.suptitle('Accuracy & Loss of Training & Validation Sets per Epoch')

    pyplot.savefig(run.accuracy)


def addModelDiagram(run):
    model = keras.models.load_model(run.model)
    keras.utils.plot_model(model, run.modelDiagram)

def addModelSummary(run):
    model = keras.models.load_model(run.model)
    with open(run.modelSummary, 'w+') as f:
        model.summary(print_fn = lambda x: f.write(x + '\n'))

if __name__ == '__main__':
    for runName in next(os.walk(path.run.directory))[1]:
        run = path.run.loadFromName(runName)
        if not path.run.has(run.accuracy):
            addAccuracyPlot(run)
        if not path.run.has(run.modelDiagram):
            addModelDiagram(run)
        addModelSummary(run)
```

Here is an example accuracy & loss plot for a training run:



Testing

```
import keras
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as pyplot
import numpy as np

import common
import preprocessing
import batch

model = load_model(modelPath)

test_generator = batch.batchGenerator.flow_from_directory(
    directory=testSetPath,
    target_size=(common.imageSize, common.imageSize),
    color_mode='grayscale',
    shuffle=True,
    batch_size=1)

filenames = test_generator.filenames
nb_samples = len(filenames)

fig=pyplot.figure()
columns = 4
rows = 4

for i in range(1, columns*rows):
    x_batch, y_batch = test_generator.next()
```

```
    name = model.predict(x_batch)
    name = np.argmax(name, axis=-1)
    true_name = y_batch
    true_name = np.argmax(true_name, axis=-1)

    label_map = (batch.trainIterator.class_indices)
    label_map = dict((v,k) for k,v in label_map.items()) #flip k,v
    predictions = [label_map[k] for k in name]
    true_value = [label_map[k] for k in true_name]

    image = x_batch[0]
    fig.add_subplot(rows, columns, i)
    pyplot.axis('off')
    pyplot.title(f"guess: {predictions[0]}\nactual: {true_value[0]}")
    pyplot.imshow(image[:,:,0], cmap='gray')

pyplot.show()
```

```
Found 31368 images belonging to 43 classes.
Found 7841 images belonging to 43 classes.
Found 2 images belonging to 1 classes.
```

Resources

Tutorial

- https://towardsdatascience.com/recognizing-traffic-signs-with-over-98-accuracy-using-deep-learning-86737aedc2ab
- https://github.com/kenshiro-o/CarND-Traffic-Sign-Classifier-Project
- https://github.com/kenshiro-o/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.ipynb

Dataset

- https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html
- http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

Loading Images

- https://www.tensorflow.org/tutorials/load_data/images