DATASET EXPLANATION

- Our goal is to build a multiclassifier model based on deep learning to classify various traffic signs.
- Dataset that we are using to train the model is **German Traffic Sign Recognition Benchmark**.
- Dataset consists of 43 classes:
- (0, b'Speed limit (20km/h)') (1, b'Speed limit (30km/h)') (2, b'Speed limit (50km/h)') (3, b'Speed limit (60km/h)') (4, b'Speed limit (70km/h)')
- (5, b'Speed limit (80km/h)') (6, b'End of speed limit (80km/h)') (7, b'Speed limit (100km/h)') (8, b'Speed limit (120km/h)') (9, b'No passing')
- (10, b'No passing for vehicles over 3.5 metric tons') (11, b'Right-of-way at the next intersection') (12, b'Priority road') (13, b'Yield') (14, b'Stop')
- (15, b'No vehicles') (16, b'Vehicles over 3.5 metric tons prohibited') (17, b'No entry')
- (18, b'General caution') (19, b'Dangerous curve to the left')
- (20, b'Dangerous curve to the right') (21, b'Double curve')
- (22, b'Bumpy road') (23, b'Slippery road')
- (24, b'Road narrows on the right') (25, b'Road work')
- (26, b'Traffic signals') (27, b'Pedestrians') (28, b'Children crossing')
- (29, b'Bicycles crossing') (30, b'Beware of ice/snow')
- (31, b'Wild animals crossing')
- (32, b'End of all speed and passing limits') (33, b'Turn right ahead')
- (34, b'Turn left ahead') (35, b'Ahead only') (36, b'Go straight or right')
- (37, b'Go straight or left') (38, b'Keep right') (39, b'Keep left')
- (40, b'Roundabout mandatory') (41, b'End of no passing')
- (42, b'End of no passing by vehicles over 3.5 metric tons')

- **Data Source** - https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

GET THE DATA AND VISUALIZE IT

In [1]:
```
import pickle

with open("train.p", mode='rb') as training_data:
    train = pickle.load(training_data)
with open("valid.p", mode='rb') as validation_data:
    valid = pickle.load(validation_data)
with open("test.p", mode='rb') as testing_data:
    test = pickle.load(testing_data)
```

In [2]:
```
X_train, y_train = train['features'], train['labels']
X_validation, y_validation = valid['features'], valid['labels']
X_test, y_test = test['features'], test['labels']
```

In [3]:
```
X_test.shape
```

Out[3]:
```
(12630, 32, 32, 3)
```

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
i = np.random.randint(1, len(X_test))
plt.imshow(X_test[i])
print('label = ', y_test[i])
```

```
label =  28
```

In [5]:
```
W_grid = 5
L_grid = 5

fig, axes = plt.subplots(L_grid, W_grid, figsize = (10,10))

axes = axes.ravel()

n_training = len(X_test)

for i in np.arange(0, W_grid * L_grid):
```

```
  File "<ipython-input-5-ca733724a555>", line 10
    for i in np.arange(0, W_grid * L_grid):
                                           ^
SyntaxError: unexpected EOF while parsing
```

IMPORT SAGEMAKER/BOTO3, CREATE A SESSION, DEFINE S3 AND ROLE

In [6]:
```
import sagemaker
import boto3


sagemaker_session = sagemaker.Session()

bucket = 'sagemaker-practical'
prefix = 'traffic-sign-classifier'

role = sagemaker.get_execution_role()
print(role)
```

```
arn:aws:iam::743472474152:role/service-role/AmazonSageMaker-ExecutionRole-20200601T134136
```

UPLOAD THE DATA TO S3

In [7]:
```
import os
os.makedirs("./data", exist_ok = True)
```

In [8]:
```
np.savez('./data/training', image = X_train, label = y_train)
np.savez('./data/validation', image = X_test, label = y_test)
```

In [9]:
```
prefix = 'traffic-sign'

training_input_path   = sagemaker_session.upload_data('data/training.npz', key_prefix = prefix + '/training')
validation_input_path = sagemaker_session.upload_data('data/validation.npz', key_prefix = prefix + '/validation')

print(training_input_path)
print(validation_input_path)
```

```
s3://sagemaker-us-east-2-743472474152/traffic-sign/training/training.npz
s3://sagemaker-us-east-2-743472474152/traffic-sign/validation/validation.npz
```

TRAIN THE CNN LENET MODEL USING SAGEMAKER

In [21]:
```
!pygmentize train-cnn.py
```

```
import argparse, os
import numpy as np
import tensorflow
from tensorflow.keras import backend as K
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization, Conv2D, MaxPooling2D, Av
eragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import multi_gpu_model


if __name__ == '__main__':

    parser = argparse.ArgumentParser()

    parser.add_argument('--epochs', type=int, default=1)
    parser.add_argument('--learning-rate', type=float, default=0.001)
    parser.add_argument('--batch-size', type=int, default=32)

    parser.add_argument('--gpu-count', type=int, default=os.environ['SM_NUM_GPUS'])
    parser.add_argument('--model-dir', type=str, default=os.environ['SM_MODEL_DIR'])
    parser.add_argument('--training', type=str, default=os.environ['SM_CHANNEL_TRAINING'])
    parser.add_argument('--validation', type=str, default=os.environ['SM_CHANNEL_VALIDATION'])

    args, _ = parser.parse_known_args()

    epochs     = args.epochs
    lr         = args.learning_rate
    batch_size = args.batch_size
    gpu_count  = args.gpu_count
    model_dir  = args.model_dir
    training_dir   = args.training
    validation_dir = args.validation

    train_images = np.load(os.path.join(training_dir, 'training.npz'))['image']
    train_labels = np.load(os.path.join(training_dir, 'training.npz'))['label']
    test_images  = np.load(os.path.join(validation_dir, 'validation.npz'))['image']
    test_labels  = np.load(os.path.join(validation_dir, 'validation.npz'))['label']

    K.set_image_data_format('channels_last')

    train_images = train_images.reshape(train_images.shape[0], 32, 32, 3)
    test_images = test_images.reshape(test_images.shape[0], 32, 32, 3)
    input_shape = (32, 32, 3)

    train_images = train_images.astype('float32')
    test_images = test_images.astype('float32')
    train_images /= 255
    test_images /= 255

    train_labels = tensorflow.keras.utils.to_categorical(train_labels, 43)
    test_labels = tensorflow.keras.utils.to_categorical(test_labels, 43)

    model = Sequential()

    model.add(Conv2D(filters=6, kernel_size=(5, 5), activation='relu', input_shape= input_shape))

    model.add(AveragePooling2D())

    model.add(Conv2D(filters=16, kernel_size=(5, 5), activation='relu'))

    model.add(AveragePooling2D())

    model.add(Dropout(0.5))

    model.add(Flatten())

    model.add(Dense(units=120, activation='relu'))

    model.add(Dropout(0.25))

    model.add(Dense(units=84, activation='relu'))

    model.add(Dense(units=43, activation = 'softmax'))

    print(model.summary())

    if gpu_count > 1:

        model = multi_gpu_model(model, gpus=gpu_count)

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

    model.fit(train_images, train_labels, batch_size=batch_size,
                  validation_data=(test_images, test_labels),
                  epochs=epochs,
```

```
                    verbose=2)

    score = model.evaluate(test_images, test_labels, verbose=0)
    print('Validation loss    :', score[0])
    print('Validation accuracy:', score[1])

    sess = K.get_session()
    tensorflow.saved_model.simple_save(
        sess,
        os.path.join(model_dir, 'model/1'),
        inputs={'inputs': model.input},
        outputs={t.name: t for t in model.outputs})
```

In [22]:
```
from sagemaker.tensorflow import TensorFlow
tf_estimator = TensorFlow(entry_point='train-cnn.py',
                          role=role,
                          train_instance_count=1,
                          train_instance_type='ml.c4.2xlarge',
                          framework_version='1.12',
                          py_version='py3',
                          script_mode=True,
                          hyperparameters={
                              'epochs': 2 ,
                              'batch-size': 32,
                              'learning-rate': 0.001}
                         )
```

In [23]:
```
tf_estimator.fit({'training': training_input_path, 'validation': validation_input_path})
```

```
None
Train on 34799 samples, validate on 12630 samples
Epoch 1/2

 - 14s - loss: 1.9968 - acc: 0.4305 - val_loss: 1.0135 - val_acc: 0.6989
Epoch 2/2
 - 13s - loss: 0.9599 - acc: 0.6958 - val_loss: 0.6868 - val_acc: 0.8029
Validation loss    : 0.686756389365721
Validation accuracy: 0.8028503563039754
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/saved_model/simple_save.py:85: calli
ng SavedModelBuilder.add_meta_graph_and_variables (from tensorflow.python.saved_model.builder_impl) with legacy_init_
op is deprecated and will be removed in a future version.
Instructions for updating:
Pass your op to the equivalent parameter main_op instead.
2020-06-01 19:30:13,961 sagemaker-containers INFO     Reporting training SUCCESS

2020-06-01 19:30:21 Uploading - Uploading generated training model
2020-06-01 19:30:21 Completed - Training job completed
Training seconds: 76
Billable seconds: 76
```

DEPLOY THE MODEL WITHOUT ACCELERATORS

In [24]:
```
import time

tf_endpoint_name = 'trafficsignclassifier-' + time.strftime("%Y-%m-%d-%H-%M-%S", time.gmtime())

tf_predictor = tf_estimator.deploy(initial_instance_count = 1,
                          instance_type = 'ml.t2.medium',
                          endpoint_name = tf_endpoint_name)
```

```
-------------!
```

In [27]:
```
%matplotlib inline
import random
import matplotlib.pyplot as plt


num_samples = 5
indices = random.sample(range(X_test.shape[0] - 1), num_samples)
images = X_test[indices]/255
labels = y_test[indices]

for i in range(num_samples):
    plt.subplot(1,num_samples,i+1)
    plt.imshow(images[i])
    plt.title(labels[i])
    plt.axis('off')

prediction = tf_predictor.predict(images.reshape(num_samples, 32, 32, 3))['predictions']
prediction = np.array(prediction)
predicted_label = prediction.argmax(axis=1)
print('Predicted labels are: {}'.format(predicted_label))
```

```
Predicted labels are: [ 4  2  6 17 12]
```

In [ ]:
```
tf_predictor.delete_endpoint()
```

In [ ]:
```

```