Context

This dataset was created by Yaroslav Bulatov by taking some publicly available fonts and extracting glyphs from them to make a dataset similar to MNIST. There are 10 classes, with letters A-J.

Content

A set of training and test images of letters from A to J on various typefaces. The images size is 28x28 pixels.

Acknowledgements

The dataset can be found on Tensorflow github page as well as on the blog from Yaroslav, here.

Inspiration

This is a pretty good dataset to train classifiers! According to Yaroslav:

Judging by the examples, one would expect this to be a harder task than MNIST. This seems to be the case -- logistic regression on top of stacked auto-encoder with fine-tuning gets about 89% accuracy whereas same approach gives got 98% on MNIST. Dataset consists of small hand-cleaned part, about 19k instances, and large uncleaned dataset, 500k instances. Two parts have approximately 0.5% and 6.5% label error rate. I got this by looking through glyphs and counting how often my guess of the letter didn't match it's unicode value in the font file. Enjoy!

In [92]:
```
import numpy as np
import tensorflow as tf
import os
import pandas as pd
```

In [93]:
```
import matplotlib.pyplot as plt
import cv2
from PIL import Image
from matplotlib import pyplot as plt
```

In [94]:
```
from sklearn.model_selection import train_test_split
```

In [95]:
```
from sklearn.preprocessing import OneHotEncoder
```

Getting the path names to images and labeling into numpy array

In [119]:
```
parentDir = './notMNIST_large/'
print(parentDir)
image_paths = []
for folder in os.listdir(parentDir):
    if folder != '.DS_Store':
        for file in os.listdir(parentDir + folder):
            image_paths.append([parentDir + folder + '/' + file, folde
r])
dataset = pd.DataFrame(image_paths)
dataset.head()
```

```
./notMNIST_large/
```

Out[119]:

	0	1
0	./notMNIST_large/I/VmFkaW0ncyBXcmI0aW5nLnR0Zg=...	I
1	./notMNIST_large/I/Q3JlZXBpbmcgRXZpbC50dGY=.png	I
2	./notMNIST_large/I/Y2FyaWNhdHVyZS50dGY=.png	I
3	./notMNIST_large/I/Q2l0eSBEIEVFIEJvbGQucGZi.png	I
4	./notMNIST_large/I/S2VwbGVyU3RkLUNuU3ViaC5vdGY...	I

In [120]:
```
print(len(dataset))
```

```
529119
```

Set batch size and epochs

Don't want batch size to be too large or not too small

In [121]:
```
batch_size = 16
num_epochs = 100
def input_func(features, labels, batch_size):

    def parser(image, label):

        img = tf.image.decode_png(tf.read_file(image))
        img = tf.image.resize_images(img, tf.constant([1, 784]))
        img = tf.reshape(img, [28, 28, 1])
        img = tf.cast(img, tf.float32, "cast")
#         image = tf.reshape(image, [28, 28, 1])
#         label = tf.one_hot(indices = label, depth = 10)
        return img, label

# #     features = tf.convert_to_tensor(data[[i for i in range(784)]])
# #     labels = tf.convert_to_tensor(pd.factorize(data['label'])[0])

#     return tf.estimator.inputs.numpy_input_fn(
#         x = {'x' : features},
#         y = labels,
#         batch_size = batch_size,
#         num_epochs = num_epochs,
#         shuffle = True
#     )


    dataset = tf.data.Dataset.from_tensor_slices((features, labels))
    dataset = dataset.map(parser)
    dataset = dataset.batch(batch_size)
    return dataset
#     feature_dict = {feature : tf.convert_to_tensor(data[feature]) fo
r feature in data if feature != 'label'}
#     labels = tf.convert_to_tensor(pd.factorize(data['label'])[0])
#     dataset = tf.data.Dataset.from_tensor_slices((feature_dict, labe
ls))
    dataset = dataset.batch(batch_size)
#     dataset = dataset.repeat(num_epochs)
    return dataset
#     iterator = dataset.make_one_shot_iterator()
#     features, labels = iterator.get_next()
#     return features, labels
```

Define model architecture

Uses a two layer, each layer consisting of a convolutional and pooling layer, architecture. (Same architecture as original MNIST CNN)

In [122]:
```
def my_model(features, labels, mode, params):
    #initialize input by reshaping and casting for network
    #img = tf.image.decode_png(tf.read_file(features['x'][0]))
    # img = np.array( img, dtype='uint8' ).flatten()

    # FIRST LAYER
    # ---conv layer with 32 filters, 5x5 kernel, and relu activation
    # ---pool layer with 2x2 pool window and stride of 2x2
    conv1 = tf.layers.conv2d(inputs=features, filters=32, kernel_size=
(5, 5), padding="same", activation=tf.nn.relu)
    pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=(2, 2), st
rides=(2, 2))

    # SECOND LAYER
    # ---conv layer with 64 filters, 5x5 kernel, and relu activation
    # ---pool layer with 2x2 pool window and stride of 2x2
    conv2 = tf.layers.conv2d(inputs=pool1, filters=64, kernel_size=(5,
5), padding="same", activation=tf.nn.relu)
    pool2 = tf.layers.max_pooling2d(inputs=conv2, pool_size=(2, 2), st
rides=(2, 2))

    # DENSE LAYER
    # ---flatten output into vector
    # ---dropout to prevent overfitting
    pool2_flat = tf.reshape(pool2, [-1, 7 * 7 * 64])
    dense = tf.layers.dense(inputs=pool2_flat, units=1024, activation=
tf.nn.relu)
    dropout = tf.layers.dropout(inputs=dense, rate=0.4, training=mode
== tf.estimator.ModeKeys.TRAIN)

    logits = tf.layers.dense(inputs=dropout, units=10)
    predictions = {
        "classes": tf.argmax(input=logits, axis=1),
        "probabilities": tf.nn.softmax(logits, name="softmax_tensor")
    }

    if mode == tf.estimator.ModeKeys.PREDICT:
        return tf.estimator.EstimatorSpec(mode=mode, predictions=predi
ctions)

    onehot_labels = tf.one_hot(indices=tf.cast(labels, tf.int32), dept
h=10)
```

```
    onehot_labels = tf.reshape(onehot_labels, [-1, 10])
    loss = tf.losses.softmax_cross_entropy(onehot_labels=onehot_labels
, logits=logits)
#     loss = tf.losses.sparse_softmax_cross_entropy(labels=labels, log
its=logits)

    if mode == tf.estimator.ModeKeys.TRAIN:
        optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.
001)
        train_op = optimizer.minimize(loss=loss,global_step=tf.train.g
et_global_step())
        return tf.estimator.EstimatorSpec(mode=mode, loss=loss, train_
op=train_op)
    print(labels.shape)
    print(predictions["classes"].shape)
    eval_metric_ops = {"accuracy": tf.metrics.accuracy(labels=labels,
predictions=predictions["classes"])}
    return tf.estimator.EstimatorSpec(mode=mode, loss=loss, eval_metri
c_ops=eval_metric_ops)
```

Training and Evaluating

-> split data into train and test (2:1) -> instantiate model with my_mode as cnn -> convert dataset (np array) to dataframe to use pd.factorize to get integer labels, then convert back to np array

In [123]:
```
# Fetch the data
X_train, X_test, y_train, y_test = train_test_split(dataset[0], pd.fac
torize(dataset[1])[0], test_size=0.33, random_state=42)

# Build CNN.
classifier = tf.estimator.Estimator(model_fn=my_model)

# Train the Model.
X_train = np.asarray(X_train)
y_train = np.asarray(y_train)

X_test = np.asarray(X_test)
y_test = np.asarray(y_test)

# print(X_train, y_train)

# train_input_func = tf.estimator.inputs.numpy_input_fn(x = {'x' : X_t
rain},
#                                                       y = y_train,
#                                                       batch_size = b
atch_size,
#                                                       num_epochs = n
um_epochs,
#                                                       shuffle = True
#                                                      )
classifier.train(input_fn=lambda:input_func(X_train, y_train, batch_si
ze), steps = 1)

# Evaluate the model.
# eval_input_func = tf.estimator.inputs.numpy_input_fn(x = {'x' : X_te
st},
#                                                      y = y_test,
#                                                      batch_size = b
atch_size,
#                                                      num_epochs = n
um_epochs,
#                                                      shuffle = True
#                                                     )
eval_result = classifier.evaluate(input_fn=lambda:input_func(X_test, y
_test, 1))

print('\nTest set accuracy: {accuracy:0.3f}\n'.format(**eval_result))
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: /var/f
olders/th/svpqqvhs62790bm9gczzcth40000gn/T/tmpeet7ux4q
INFO:tensorflow:Using config: {'_service': None, '_task_type': 'work
er', '_keep_checkpoint_every_n_hours': 10000, '_num_ps_replicas': 0,
'_num_worker_replicas': 1, '_master': '', '_global_id_in_cluster': 0
, '_log_step_count_steps': 100, '_save_summary_steps': 100, '_train_
distribute': None, '_cluster_spec': <tensorflow.python.training.serv
er_lib.ClusterSpec object at 0x1a2c7c7f60>, '_is_chief': True, '_tf_
random_seed': None, '_save_checkpoints_secs': 600, '_evaluation_mast
er': '', '_keep_checkpoint_max': 5, '_session_config': None, '_task_
id': 0, '_save_checkpoints_steps': None, '_model_dir': '/var/folders
/th/svpqqvhs62790bm9gczzcth40000gn/T/tmpeet7ux4q'}
WARNING:tensorflow:Estimator's model_fn (<function my_model at 0x1a2
823ee18>) includes params argument, but params are not passed to Est
imator.
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 1 into /var/folders/th/svpqqv
hs62790bm9gczzcth40000gn/T/tmpeet7ux4q/model.ckpt.
INFO:tensorflow:step = 1, loss = 20.534435
INFO:tensorflow:Loss for final step: 20.534435.
INFO:tensorflow:Calling model_fn.
(?,)
(?,)
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2018-06-22-05:00:06
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /var/folders/th/svpqqvhs62
790bm9gczzcth40000gn/T/tmpeet7ux4q/model.ckpt-1
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.

---------------------------------------------------------------------
-------
InvalidArgumentError                      Traceback (most recent cal
l last)
~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/c
lient/session.py in _do_call(self, fn, *args)
   1321     try:
-> 1322       return fn(*args)
   1323     except errors.OpError as e:

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/c
lient/session.py in _run_fn(feed_dict, fetch_list, target_list, opti
ons, run_metadata)
```

```
   1306       return self._call_tf_sessionrun(
-> 1307           options, feed_dict, fetch_list, target_list, run_m
etadata)
   1308

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/c
lient/session.py in _call_tf_sessionrun(self, options, feed_dict, fe
tch_list, target_list, run_metadata)
   1408           self._session, options, feed_dict, fetch_list,
target_list,
-> 1409           run_metadata)
   1410     else:

InvalidArgumentError: Expected image (JPEG, PNG, or GIF), got empty
file
         [[Node: DecodePng = DecodePng[channels=0, dtype=DT_UINT8](R
eadFile)]]
         [[Node: IteratorGetNext = IteratorGetNext[output_shapes=[[?
,28,28,1], [?]], output_types=[DT_FLOAT, DT_INT64], _device="/job:lo
calhost/replica:0/task:0/device:CPU:0"](Iterator)]]

During handling of the above exception, another exception occurred:

InvalidArgumentError                      Traceback (most recent cal
l last)
<ipython-input-123-14d1fea70048> in <module>()
     29 #                                                          shuf
fle = True
     30 #                                                         )
---> 31 eval_result = classifier.evaluate(input_fn=lambda:input_func
(X_test, y_test, 1))
     32
     33 print('\nTest set accuracy: {accuracy:0.3f}\n'.format(**eval
_result))

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/e
stimator/estimator.py in evaluate(self, input_fn, steps, hooks, chec
kpoint_path, name)
    423         hooks=hooks,
    424         checkpoint_path=checkpoint_path,
--> 425         name=name)
    426
    427   def _convert_eval_steps_to_hooks(self, steps):

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/e
stimator/estimator.py in _evaluate_model(self, input_fn, hooks, chec
kpoint_path, name)
   1115           final_ops=eval_dict,
   1116           hooks=all_hooks,
-> 1117           config=self._session_config)
```

```
   1118
   1119       _write_dict_to_summary(

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/t
raining/evaluation.py in _evaluate_once(checkpoint_path, master, sca
ffold, eval_ops, feed_dict, final_ops, final_ops_feed_dict, hooks, c
onfig)
    210     if eval_ops is not None:
    211       while not session.should_stop():
--> 212         session.run(eval_ops, feed_dict)
    213
    214   logging.info('Finished evaluation at ' + time.strftime('%Y
-%m-%d-%H:%M:%S',

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/t
raining/monitored_session.py in run(self, fetches, feed_dict, option
s, run_metadata)
    565                           feed_dict=feed_dict,
    566                           options=options,
--> 567                           run_metadata=run_metadata)
    568
    569   def run_step_fn(self, step_fn):

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/t
raining/monitored_session.py in run(self, fetches, feed_dict, option
s, run_metadata)
   1041                                 feed_dict=feed_dict,
   1042                                 options=options,
-> 1043                                 run_metadata=run_metadata)
   1044       except _PREEMPTION_ERRORS as e:
   1045         logging.info('An error was raised. This may be due t
o a preemption in '

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/t
raining/monitored_session.py in run(self, *args, **kwargs)
   1132         raise six.reraise(*original_exc_info)
   1133       else:
-> 1134         raise six.reraise(*original_exc_info)
   1135
   1136

~/anaconda3/envs/py3/lib/python3.5/site-packages/six.py in reraise(t
p, value, tb)
    691             if value.__traceback__ is not tb:
    692                 raise value.with_traceback(tb)
--> 693             raise value
    694         finally:
    695             value = None

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/t
```

```
raining/monitored_session.py in run(self, *args, **kwargs)
   1117   def run(self, *args, **kwargs):
   1118     try:
-> 1119       return self._sess.run(*args, **kwargs)
   1120     except _PREEMPTION_ERRORS:
   1121       raise

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/t
raining/monitored_session.py in run(self, fetches, feed_dict, option
s, run_metadata)
   1189                                     feed_dict=feed_dict,
   1190                                     options=options,
-> 1191                                     run_metadata=run_metadata)
   1192
   1193     for hook in self._hooks:

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/t
raining/monitored_session.py in run(self, *args, **kwargs)
    969
    970   def run(self, *args, **kwargs):
--> 971     return self._sess.run(*args, **kwargs)
    972
    973   def run_step_fn(self, step_fn, raw_session, run_with_hooks
):

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/c
lient/session.py in run(self, fetches, feed_dict, options, run_metad
ata)
    898     try:
    899       result = self._run(None, fetches, feed_dict, options_p
tr,
--> 900                          run_metadata_ptr)
    901       if run_metadata:
    902         proto_data = tf_session.TF_GetBuffer(run_metadata_pt
r)

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/c
lient/session.py in _run(self, handle, fetches, feed_dict, options,
run_metadata)
   1133     if final_fetches or final_targets or (handle and
feed_dict_tensor):
   1134       results = self._do_run(handle, final_targets, final_fe
tches,
-> 1135                              feed_dict_tensor, options, run_
metadata)
   1136     else:
   1137       results = []

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/c
lient/session.py in _do_run(self, handle, target_list, fetch_list, f
```

```
eed_dict, options, run_metadata)
   1314     if handle is None:
   1315       return self._do_call(_run_fn, feeds, fetches, targets,
options,
-> 1316                            run_metadata)
   1317     else:
   1318       return self._do_call(_prun_fn, handle, feeds, fetches)

~/anaconda3/envs/py3/lib/python3.5/site-packages/tensorflow/python/c
lient/session.py in _do_call(self, fn, *args)
   1333         except KeyError:
   1334           pass
-> 1335       raise type(e)(node_def, op, message)
   1336
   1337   def _extend_graph(self):

InvalidArgumentError: Expected image (JPEG, PNG, or GIF), got empty
file
         [[Node: DecodePng = DecodePng[channels=0, dtype=DT_UINT8](R
eadFile)]]
         [[Node: IteratorGetNext = IteratorGetNext[output_shapes=[[?
,28,28,1], [?]], output_types=[DT_FLOAT, DT_INT64], _device="/job:lo
calhost/replica:0/task:0/device:CPU:0"](Iterator)]]
```

In [72]:
```
print(X_test, len(X_test))
print(y_test, len(y_test))
```

```
['notMNIST_large/A/VW5pdmVyc0xULUV4dHJhQmxhY2tFeHQub3Rm.png'
 'notMNIST_large/F/Q3JheW9uIE5vcm1hbC50dGY=.png'
 'notMNIST_large/B/QnVyb2tyYXQtT25lLm90Zg==.png' ...
 'notMNIST_large/E/SW5zdGFsbGF0aW9uIFNTaSBCb2xkLnR0Zg==.png'
 'notMNIST_large/H/QmF1ZXIgQm9kb25pIEl0YWxpYy5wZmI=.png'
 'notMNIST_large/I/VHJpYW5nZWwudHRm.png'] 174610
[2 3 9 ... 8 4 0] 174610
```

In [73]:
```
print(X_train, len(X_train))
print(y_train, len(y_train))
```

```
['notMNIST_large/F/SGFsbGFuZGFsZSBTQyBCb2xkIEl0LiBKTC50dGY=.png'
 'notMNIST_large/G/UmFndGltZVN0ZC5vdGY=.png'
 'notMNIST_large/H/SGFycmluZ3RvbiBSZWd1bGFyLnR0Zg==.png' ...
 'notMNIST_large/C/RkZEaW5nYmF0cy1OdW1iZXIub3Rm.png'
 'notMNIST_large/A/UGhvdGluYSBNVCBTZW1pIEJvbGQgSXRhbGljLnR0Zg==.png'
 'notMNIST_large/A/V2VsbHJvY2tTbGFiLnR0Zg==.png'] 354509
[3 1 4 ... 6 2 2] 354509
```