Создание сверточной сети и использование MobileNet для распознование номеров домов с количеством цифр в номере от 1-го до 6

```
import os
try:
  import wget
except:
  !pip install wget
  import wget
import tarfile


out_dir = 'data/svhn'

train_32_32 = ('http://ufldl.stanford.edu/housenumbers/train_32x32.mat', 'train_32x32.mat')
test_32_32 = ('http://ufldl.stanford.edu/housenumbers/test_32x32.mat', 'test_32x32.mat')
extra_32_32 = ('http://ufldl.stanford.edu/housenumbers/extra_32x32.mat', 'extra_32x32.mat')

train_large = ('http://ufldl.stanford.edu/housenumbers/train.tar.gz', 'train.tar.gz')
test_large = ('http://ufldl.stanford.edu/housenumbers/test.tar.gz', 'test.tar.gz')
extra_large = ('http://ufldl.stanford.edu/housenumbers/extra.tar.gz', 'extra.tar.gz')


import tensorflow as tf
tf.test.gpu_device_name()
```

```
'/device:GPU:0'
```

```
def download_data(url, filename, out_dir=out_dir):
    filename = os.path.join(out_dir, filename)

    if not os.path.exists(out_dir):
        os.makedirs(out_dir)

    if not os.path.exists(filename):
        print(f"Downloading {filename}.")
        wget.download(url, filename)
```

```
            print()
        else:
            print(f"Skipping {filename} download (already exists)")


    def extract_data(filename, out_dir=out_dir):
        filename = os.path.join(out_dir, filename)

        print(f"Extracting {filename}")
        with tarfile.open(filename) as tar:
            tar.extractall(out_dir)

    download_data(*train_32_32)
    download_data(*test_32_32)
    download_data(*extra_32_32)

    download_data(*train_large)
    download_data(*test_large)
    download_data(*extra_large)

    extract_data(train_large[1])
    extract_data(test_large[1])
    extract_data(extra_large[1])
```

```
Skipping data/svhn/train_32x32.mat download (already exists)
Skipping data/svhn/test_32x32.mat download (already exists)
Skipping data/svhn/extra_32x32.mat download (already exists)
Skipping data/svhn/train.tar.gz download (already exists)
Skipping data/svhn/test.tar.gz download (already exists)
Downloading data/svhn/extra.tar.gz.

Extracting data/svhn/train.tar.gz
Extracting data/svhn/test.tar.gz
Extracting data/svhn/extra.tar.gz
```

```
# -*- coding: utf-8 -*-
from tensorflow import keras
import numpy as np
```

```
import numpy as np
from PIL import Image
from pathlib import Path
from scipy import io
import h5py
import json

def to_one_hot(a, n):
    result = np.zeros(shape=(a.shape[0], n))
    result[np.arange(len(a)), a] = 1
    return result

def load_multiple_digits_data(dir='data/svhn', train=True, extra=False):

    def parse_digit_struct(file):
        print('file - ' + str(file))
        if Path(f"{file}.cache.json").exists() and os.stat(f"{file}.cache.json").st_size != 0:
            print('exist')
            with open(f"{file}.cache.json", "r") as f:
                images = json.load(f)
                print(f'Loaded cached image attrs from {file}.cache.json')
                return images

        f = h5py.File(file, 'r')
        print(f'Opened file {file}')

        names = f['digitStruct']['name']
        bbox = f['digitStruct']['bbox']

        def extract_name(i):
            return ''.join([chr(c[0]) for c in f[names[i][0]].value])

        def extract_attr(i, attr):
            attr = f[bbox[i].item()][attr]
            if len(attr) > 1:
                return [f[attr.value[j].item()].value[0][0] for j in range(len(attr))]
            else:
                return [attr.value[0][0]]
```

```
        images = {}
        print(f'Extracting image attrs from {file}: ', end='')
        for i in range(len(names)):
            name = extract_name(i)
            images[name] = {
                "label": extract_attr(i, 'label'),
                "top": extract_attr(i, 'top'),
                "left": extract_attr(i, 'left'),
                "height": extract_attr(i, 'height'),
                "width": extract_attr(i, 'width')
            }
            if i % 1000 == 0:
                print('.', end='', flush=True)
        print()

        with open(f"{file}.cache.json", 'w+') as f:
            json.dump(images, f)
        return images

    def process_images(dir):
        cache_file = Path(dir) / 'cache.npz'
        if cache_file.exists():
            f = np.load(cache_file)
            print(f'Loaded cached arrays for {dir}')
            return [v for k, v in f.items()]

        attrs = parse_digit_struct(Path(dir) / 'digitStruct.mat')

        x, y = [], []
        print(f'Processing images from {dir}: ', end='', flush=True)
        for i, name in enumerate(os.listdir(dir)):
            if name not in attrs:
                print('s', end='', flush=True)
                continue

            img = Image.open(Path(dir) / name)
```

```
            height = int(max(attrs[name]['height']))
            width = int(max(attrs[name]['width']))
            left = max(int(min(attrs[name]['left'])) - 0.5 * width, 0)
            top = max(int(min(attrs[name]['top'])) - 0.5 * height, 0)
            right = min(int(max(attrs[name]['left'])) + 1.5 * width, img.size[0])
            bottom = min(int(max(attrs[name]['top'])) + 1.5 * height, img.size[1])

            img = img.crop(box=(left, top, right, bottom))
            img = img.resize((96, 96))

            label = [d % 10 for d in attrs[name]['label']]
            if len(label) > 6:
                print('e', end='', flush=True)
                continue

            label += [10] * (6 - len(label))
            label = to_one_hot(np.array(label, dtype=np.int), 11)

            x.append(np.array(img))
            y.append(np.array(label))

            if i % 1000 == 0:
                print('.', end='', flush=True)
        print()

        x = np.array(x, dtype=np.uint8)
        y = np.array(y, dtype=np.uint8)
        np.savez(Path(dir) / "cache.npz", x, y)
        return x, y

    x_test, y_test = process_images(Path(dir) / 'test/')

    x_train, y_train = None, None
    if train:
        x_train, y_train = process_images(Path(dir) / 'train/')

    x_extra, y_extra = None, None
    if extra:
```

```
        x_extra, y_extra = process_images(Path(dir) / 'extra/')

    return (
        x_train, y_train,
        x_test, y_test,
        x_extra, y_extra
    )

    x_test, y_test = process_images(Path(dir) / 'test/')

    x_train, y_train = None, None
    if train:
        x_train, y_train = process_images(Path(dir) / 'train/')

    x_extra, y_extra = None, None
    if extra:
        x_extra, y_extra = process_images(Path(dir) / 'extra/')

    return (
        x_train, y_train,
        x_test, y_test,
        x_extra, y_extra
    )
```

```
x_train, y_train, x_test, y_test, _, _ = load_multiple_digits_data()
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.1)

def to_y(a, n):
    return [a[:,i,:] for i in range(n)]

if y_train is not None and y_val is not None and y_test is not None:
            y_train = to_y(y_train, 6)
            y_val = to_y(y_val, 6)
            y_test = to_y(y_test, 6)
```

```
y_test = to_y(y_test, 6)
```

```
Loaded cached arrays for data/svhn/test
Loaded cached arrays for data/svhn/train
```

```
y_train[1].shape
```

```
(30061, 11)
```

```
import tensorflow as tf

from tensorflow import keras

input = keras.layers.Input(shape=(96, 96, 3))

x = keras.layers.Conv2D(16, 5, activation='relu', padding='same')(input)
x = keras.layers.MaxPool2D(pool_size=(2, 2), padding = 'same')(x)
x = keras.layers.Conv2D(32, 5, activation='relu', padding='same')(x)
x = keras.layers.MaxPool2D(pool_size=(2, 2), padding='same')(x)
x = keras.layers.Conv2D(64, 5, activation='relu', padding='same')(x)
x = keras.layers.MaxPool2D(pool_size=(2, 2), padding='same')(x)
x = keras.layers.Flatten()(x)
x = keras.layers.Dropout(rate=0.1)(x)
x = keras.layers.Dense(100, activation='relu')(x)
x = keras.layers.Dropout(rate=0.1)(x)

out1 = keras.layers.Dense(11,  activation='linear')(x)
out2 = keras.layers.Dense(11,  activation='linear')(x)
out3 = keras.layers.Dense(11,  activation='linear')(x)
out4 = keras.layers.Dense(11,  activation='linear')(x)
out5 = keras.layers.Dense(11,  activation='linear')(x)
out6 = keras.layers.Dense(11,  activation='linear')(x)


outputs = [
   keras.layers.Dense(11, activation='softmax', name=f'out_{i}')(dropout)
   for i in range(6)
```

```
]

model = keras.models.Model(
   inputs=[input],
   outputs=[out1,out2,out3,out4,out5,out6]
)
model.compile(
   optimizer=keras.optimizers.Adam(lr=0.001),
   loss='categorical_crossentropy',
   metrics=['categorical_accuracy']#,
   #loss_weights=[1, 1, 0.5, 0.3, 0.1, 0.05]
)
model.summary()
```

```
Model: "model_12"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_15 (InputLayer)           [(None, 96, 96, 3)]  0
__________________________________________________________________________________________________
conv2d_21 (Conv2D)              (None, 96, 96, 16)   1216        input_15[0][0]
__________________________________________________________________________________________________
max_pooling2d_21 (MaxPooling2D) (None, 48, 48, 16)   0           conv2d_21[0][0]
__________________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, 48, 48, 32)   12832       max_pooling2d_21[0][0]
__________________________________________________________________________________________________
max_pooling2d_22 (MaxPooling2D) (None, 24, 24, 32)   0           conv2d_22[0][0]
__________________________________________________________________________________________________
conv2d_23 (Conv2D)              (None, 24, 24, 64)   51264       max_pooling2d_22[0][0]
__________________________________________________________________________________________________
max_pooling2d_23 (MaxPooling2D) (None, 12, 12, 64)   0           conv2d_23[0][0]
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 9216)         0           max_pooling2d_23[0][0]
__________________________________________________________________________________________________
dropout_23 (Dropout)            (None, 9216)         0           flatten_6[0][0]
__________________________________________________________________________________________________
dense_26 (Dense)                (None, 100)          921700      dropout_23[0][0]
__________________________________________________________________________________________________
dropout_24 (Dropout)            (None, 100)          0           dense_26[0][0]
__________________________________________________________________________________________________
dense_27 (Dense)                (None, 11)           1111        dropout_24[0][0]
__________________________________________________________________________________________________
dense_28 (Dense)                (None, 11)           1111        dropout_24[0][0]
__________________________________________________________________________________________________
dense_29 (Dense)                (None, 11)           1111        dropout_24[0][0]
__________________________________________________________________________________________________
dense_30 (Dense)                (None, 11)           1111        dropout_24[0][0]
__________________________________________________________________________________________________
dense_31 (Dense)                (None, 11)           1111        dropout_24[0][0]
__________________________________________________________________________________________________
dense_32 (Dense)                (None, 11)           1111        dropout_24[0][0]
==================================================================================================
Total params: 993,678
Trainable params: 993,678
Non-trainable params: 0
__________________________________________________________________________________________________
```

```
model.fit(
                x_train,
                y_train,
                epochs=100,
                batch_size=32,
                verbose=2,
                validation_split=0.1,
                callbacks=[
                    keras.callbacks.EarlyStopping(
                        patience=10,
                        restore_best_weights=True
                    )
                ]
            )
```

```
Epoch 1/100
846/846 - 12s - loss: 25.5872 - dense_20_loss: 7.9882 - dense_21_loss: 8.1141 - dense_22_loss: 8.4577 - dense_23_loss
Epoch 2/100
846/846 - 11s - loss: 25.4986 - dense_20_loss: 7.7880 - dense_21_loss: 7.9509 - dense_22_loss: 9.0528 - dense_23_loss
Epoch 3/100
846/846 - 11s - loss: 24.8798 - dense_20_loss: 8.0329 - dense_21_loss: 8.3661 - dense_22_loss: 8.0816 - dense_23_loss
Epoch 4/100
846/846 - 11s - loss: 22.7401 - dense_20_loss: 7.3277 - dense_21_loss: 7.9836 - dense_22_loss: 5.6523 - dense_23_loss
Epoch 5/100
846/846 - 11s - loss: 26.9416 - dense_20_loss: 8.5300 - dense_21_loss: 8.8952 - dense_22_loss: 8.8231 - dense_23_loss
Epoch 6/100
846/846 - 11s - loss: 24.5093 - dense_20_loss: 8.6035 - dense_21_loss: 7.4870 - dense_22_loss: 6.2798 - dense_23_loss
Epoch 7/100
846/846 - 11s - loss: 23.8692 - dense_20_loss: 7.8035 - dense_21_loss: 7.5713 - dense_22_loss: 5.4336 - dense_23_loss
Epoch 8/100
846/846 - 11s - loss: 27.4052 - dense_20_loss: 8.1056 - dense_21_loss: 8.2158 - dense_22_loss: 9.8477 - dense_23_loss
Epoch 9/100
846/846 - 11s - loss: 26.9968 - dense_20_loss: 8.5868 - dense_21_loss: 8.2002 - dense_22_loss: 9.4175 - dense_23_loss
Epoch 10/100
846/846 - 11s - loss: 27.9577 - dense_20_loss: 11.6004 - dense_21_loss: 7.9561 - dense_22_loss: 7.4754 - dense_23_loss
Epoch 11/100
846/846 - 11s - loss: 23.4535 - dense_20_loss: 6.9473 - dense_21_loss: 7.6929 - dense_22_loss: 7.3438 - dense_23_loss
Epoch 12/100
846/846 - 11s - loss: 24.2445 - dense_20_loss: 4.2911 - dense_21_loss: 7.7647 - dense_22_loss: 12.2129 - dense_23_loss
Epoch 13/100
846/846 - 11s - loss: 21.5637 - dense_20_loss: 4.2988 - dense_21_loss: 7.6929 - dense_22_loss: 11.4577 - dense_23_loss
Epoch 14/100
846/846 - 11s - loss: 16.3488 - dense_20_loss: 4.2700 - dense_21_loss: 7.6798 - dense_22_loss: 2.9043 - dense_23_loss
Epoch 15/100
846/846 - 12s - loss: 16.9035 - dense_20_loss: 4.2855 - dense_21_loss: 7.6874 - dense_22_loss: 3.0038 - dense_23_loss
Epoch 16/100
846/846 - 11s - loss: 20.2126 - dense_20_loss: 4.2990 - dense_21_loss: 7.6977 - dense_22_loss: 9.5748 - dense_23_loss
Epoch 17/100
846/846 - 12s - loss: 19.7913 - dense_20_loss: 4.2936 - dense_21_loss: 7.6547 - dense_22_loss: 10.6004 - dense_23_loss
Epoch 18/100
846/846 - 11s - loss: 20.3894 - dense_20_loss: 4.9943 - dense_21_loss: 7.7964 - dense_22_loss: 9.5294 - dense_23_loss
Epoch 19/100
846/846 - 11s - loss: 22.6355 - dense_20_loss: 5.1695 - dense_21_loss: 8.1439 - dense_22_loss: 13.1791 - dense_23_loss
Epoch 20/100
846/846 - 11s - loss: 22.5832 - dense_20_loss: 5.1632 - dense_21_loss: 8.1420 - dense_22_loss: 13.3382 - dense_23_loss
Epoch 21/100
846/846 - 11s - loss: 22.0678 - dense_20_loss: 5.2079 - dense_21_loss: 8.1287 - dense_22_loss: 13.3311 - dense_23_loss
```

```
Epoch 22/100
846/846 - 11s - loss: 22.6441 - dense_20_loss: 5.1982 - dense_21_loss: 8.1474 - dense_22_loss: 13.3379 - dense_23_loss
Epoch 23/100
846/846 - 11s - loss: 23.9340 - dense_20_loss: 5.4061 - dense_21_loss: 8.1614 - dense_22_loss: 13.3302 - dense_23_loss
<tensorflow.python.keras.callbacks.History at 0x7fa2093aecc0>
```

```
y_pred = model.predict(x_test)

total = np.array([True] * len(x_test))
for i, (y1, y2) in enumerate(zip(y_test, y_pred)):
  cur = np.argmax(y1, axis=1) == np.argmax(y2, axis=1)
  total = np.logical_and(total, cur)

  acc = np.mean(cur.astype(np.int))
  #history[f'test_out_{i}_acc'] = acc
  print(f'Accuracy of out_{i} = {acc:.5f}')

acc = np.mean(total.astype(np.int))
#history['test_acc'] = acc
print(f'Accuracy = {acc:.5f}')
```

```
Accuracy of out_0 = 0.00145
Accuracy of out_1 = 0.07048
Accuracy of out_2 = 0.01852
Accuracy of out_3 = 0.00122
Accuracy of out_4 = 0.99985
Accuracy of out_5 = 0.00000
Accuracy = 0.00000
```

```
!ls data/svhn/
```

```
extra_32x32.mat  test_32x32.mat  train            train.tar.gz
test             test.tar.gz     train_32x32.mat
```

```
_, _, x_test, y_test, x_train, y_train = load_multiple_digits_data(extra=True, train=False)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.1)
```

```
if y_train is not None and y_val is not None and y_test is not None:
            y_train = to_y(y_train, 6)
            y_val = to_y(y_val, 6)
            y_test = to_y(y_test, 6)
```

```
Loaded cached arrays for data/svhn/test
file - data/svhn/extra/digitStruct.mat
Opened file data/svhn/extra/digitStruct.mat
Extracting image attrs from data/svhn/extra/digitStruct.mat: ./usr/local/lib/python3.6/dist-packages/ipykernel_launche
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:37: H5pyDeprecationWarning: dataset.value has been depre
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:39: H5pyDeprecationWarning: dataset.value has been depre
.....................................................................................................
Processing images from data/svhn/extra: ..........................................................
```

```
model.fit(
                x_train,
                y_train,
                epochs=100,
                batch_size=32,
                verbose=2,
                validation_split=0.1,
                callbacks=[
                    keras.callbacks.EarlyStopping(
                        patience=10,
                        restore_best_weights=True
                    )
                ]
            )
```

```
Epoch 1/100
5123/5123 - 67s - loss: 62.7917 - dense_27_loss: 8.3268 - dense_28_loss: 8.1887 - dense_29_loss: 6.6918 - dense_30_los
Epoch 2/100
5123/5123 - 66s - loss: 33.3925 - dense_27_loss: 8.0147 - dense_28_loss: 8.3876 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 3/100
5123/5123 - 66s - loss: 33.3926 - dense_27_loss: 8.0147 - dense_28_loss: 8.3876 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 4/100
5123/5123 - 65s - loss: 33.3956 - dense_27_loss: 8.0177 - dense_28_loss: 8.3876 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 5/100
5123/5123 - 66s - loss: 33.3926 - dense_27_loss: 8.0147 - dense_28_loss: 8.3876 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 6/100
5123/5123 - 65s - loss: 33.3895 - dense_27_loss: 8.0147 - dense_28_loss: 8.3846 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 7/100
5123/5123 - 66s - loss: 33.3895 - dense_27_loss: 8.0147 - dense_28_loss: 8.3846 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 8/100
5123/5123 - 66s - loss: 33.3925 - dense_27_loss: 8.0177 - dense_28_loss: 8.3846 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 9/100
5123/5123 - 66s - loss: 33.3895 - dense_27_loss: 8.0147 - dense_28_loss: 8.3846 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 10/100
5123/5123 - 65s - loss: 33.3956 - dense_27_loss: 8.0177 - dense_28_loss: 8.3876 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 11/100
5123/5123 - 66s - loss: 33.3925 - dense_27_loss: 8.0177 - dense_28_loss: 8.3846 - dense_29_loss: 1.1921e-07 - dense_3(
<tensorflow.python.keras.callbacks.History at 0x7fa2093f77b8>
```

```
y_pred = model.predict(x_test)

total = np.array([True] * len(x_test))
for i, (y1, y2) in enumerate(zip(y_test, y_pred)):
  cur = np.argmax(y1, axis=1) == np.argmax(y2, axis=1)
  total = np.logical_and(total, cur)

  acc = np.mean(cur.astype(np.int))
  #history[f'test_out_{i}_acc'] = acc
  print(f'Accuracy of out_{i} = {acc:.5f}')

acc = np.mean(total.astype(np.int))
#history['test_acc'] = acc
print(f'Accuracy = {acc: 5f}')
```

```
print('Accuracy = {acc:.5f}')
```

```
Accuracy of out_0 = 0.12435
Accuracy of out_1 = 0.07859
Accuracy of out_2 = 0.02265
Accuracy of out_3 = 0.98867
Accuracy of out_4 = 0.00000
Accuracy of out_5 = 0.00000
Accuracy = 0.00000
```

```
input = keras.layers.Input(shape=(96, 96, 3))

mobile_net = keras.applications.mobilenet_v2.MobileNetV2(
  include_top=False,
  weights='imagenet',
  input_shape=(96, 96, 3),
  input_tensor=input,
  pooling='avg'
)

dropout = keras.layers.Dropout(rate=0.1)(mobile_net.output)

outputs = [
  keras.layers.Dense(11, activation='softmax', name=f'out_{i}')(dropout)
  for i in range(6)
]

model1 = keras.models.Model(
  inputs=[input],
  outputs=outputs
)
model1.compile(
  optimizer=keras.optimizers.Adam(lr=0.001),
  loss='categorical_crossentropy',
  metrics=['categorical_accuracy'],
  loss_weights=[1, 1, 0.5, 0.3, 0.1, 0.05]
)
```

```
model.fit(
                x_train,
                y_train,
                epochs=100,
                batch_size=32,
                verbose=2,
                validation_split=0.1,
                callbacks=[
                    keras.callbacks.EarlyStopping(
                        patience=10,
                        restore_best_weights=True
                    )
                ]
            )
```

```
Epoch 1/100
5123/5123 - 66s - loss: 33.3926 - dense_27_loss: 8.0177 - dense_28_loss: 8.3846 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 2/100
5123/5123 - 66s - loss: 33.3925 - dense_27_loss: 8.0147 - dense_28_loss: 8.3876 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 3/100
5123/5123 - 66s - loss: 33.3896 - dense_27_loss: 8.0147 - dense_28_loss: 8.3846 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 4/100
5123/5123 - 66s - loss: 33.3895 - dense_27_loss: 8.0147 - dense_28_loss: 8.3846 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 5/100
5123/5123 - 66s - loss: 33.3925 - dense_27_loss: 8.0177 - dense_28_loss: 8.3846 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 6/100
5123/5123 - 66s - loss: 33.3956 - dense_27_loss: 8.0177 - dense_28_loss: 8.3876 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 7/100
5123/5123 - 67s - loss: 33.3956 - dense_27_loss: 8.0177 - dense_28_loss: 8.3876 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 8/100
5123/5123 - 66s - loss: 33.3925 - dense_27_loss: 8.0147 - dense_28_loss: 8.3876 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 9/100
5123/5123 - 66s - loss: 33.3926 - dense_27_loss: 8.0177 - dense_28_loss: 8.3846 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 10/100
5123/5123 - 66s - loss: 33.3926 - dense_27_loss: 8.0147 - dense_28_loss: 8.3876 - dense_29_loss: 1.1921e-07 - dense_3(
Epoch 11/100
5123/5123 - 66s - loss: 33.3926 - dense_27_loss: 8.0147 - dense_28_loss: 8.3876 - dense_29_loss: 1.1921e-07 - dense_3(
<tensorflow.python.keras.callbacks.History at 0x7fa1fe1d9470>
```

```
y_pred = model1.predict(x_test)

total = np.array([True] * len(x_test))
for i, (y1, y2) in enumerate(zip(y_test, y_pred)):
  cur = np.argmax(y1, axis=1) == np.argmax(y2, axis=1)
  total = np.logical_and(total, cur)

  acc = np.mean(cur.astype(np.int))
  #history[f'test_out_{i}_acc'] = acc
  print(f'Accuracy of out_{i} = {acc:.5f}')

acc = np.mean(total.astype(np.int))
#history['test_acc'] = acc
print(f'Accuracy = {acc:.5f}')
```

```
Accuracy of out_0 = 0.04982
Accuracy of out_1 = 0.08509
Accuracy of out_2 = 0.82744
Accuracy of out_3 = 0.00069
Accuracy of out_4 = 0.00000
Accuracy of out_5 = 0.85889
Accuracy = 0.00000
```

```
model.save_weights('models/svhn_multiple_mobile_net_extra/model')
```

```
!ls models/svhn_multiple_mobile_net_extra/
```

```
checkpoint  model.data-00000-of-00002  model.data-00001-of-00002  model.index
```

```
from google.colab import files

files.download('models/svhn_multiple_mobile_net_extra/checkpoint')
files.download('models/svhn_multiple_mobile_net_extra/model.data-00000-of-00002')
files.download('models/svhn_multiple_mobile_net_extra/model.data-00001-of-00002')
files.download('models/svhn_multiple_mobile_net_extra/model.index')
```