svhn_prediction

September 22, 2020

1 Import moduls

[63]:
```
import tensorflow as tf
assert tf.__version__ == '2.3.0', "tf version not 2.3.0"
import tensorflow_datasets as tfds
from tensorflow.keras import Sequential
from tensorflow.keras.layers import (Dense, Flatten, Conv2D, MaxPool2D,
                                     GlobalMaxPool2D, Dropout,␣
 ↪BatchNormalization,
                                     Activation)
# import callbacks
from tensorflow.keras.callbacks import (ModelCheckpoint, EarlyStopping,␣
 ↪TerminateOnNaN,
                                  ReduceLROnPlateau, TensorBoard,␣
 ↪LearningRateScheduler)

%load_ext tensorboard

import numpy as np
import matplotlib.pyplot as plt
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

2 Download dataset from tfds

[64]:
```
(train_ds, test_ds), info = tfds.load('svhn_cropped', split=['train', 'test'],
                                      with_info=True, shuffle_files=True,
                                      as_supervised=True)
train_ds, test_ds, info
```

[64]:
```
(<DatasetV1Adapter shapes: ((32, 32, 3), ()), types: (tf.uint8, tf.int64)>,
 <DatasetV1Adapter shapes: ((32, 32, 3), ()), types: (tf.uint8, tf.int64)>,
 tfds.core.DatasetInfo(
     name='svhn_cropped',
     version=3.0.0,
     description='The Street View House Numbers (SVHN) Dataset is an image digit
```

```
recognition dataset of over 600,000 digit images coming from real world data.
Images are cropped to 32x32.',
     homepage='http://ufldl.stanford.edu/housenumbers/',
     features=FeaturesDict({
         'image': Image(shape=(32, 32, 3), dtype=tf.uint8),
         'label': ClassLabel(shape=(), dtype=tf.int64, num_classes=10),
     }),
     total_num_examples=630420,
     splits={
         'extra': 531131,
         'test': 26032,
         'train': 73257,
     },
     supervised_keys=('image', 'label'),
     citation="""@article{Netzer2011,
     author = {Netzer, Yuval and Wang, Tao and Coates, Adam and Bissacco,
Alessandro and Wu, Bo and Ng, Andrew Y},
     booktitle = {Advances in Neural Information Processing Systems ({NIPS})},
     title = {Reading Digits in Natural Images with Unsupervised Feature
Learning},
     year = {2011}
     }""",
     redistribution_info=,
 ))
```

3 Create major constants

[65]:
```
cp_path_every_epoch = "checkpoint_every_epoch/"
cp_path_best_epoch = "checkpoint_best_epoch/"
log_dir = "logs_fit"

img_shape = tuple(train_ds.output_shapes[0])
img_shape
```

[65]:
```
(32, 32, 3)
```

4 Data preview

[66]:
```
# help(train_ds)
plt.figure(figsize=(8, 3.5))

np_ds = tfds.as_numpy(train_ds)
for (image, label), ii in zip(np_ds, [i for i in range(10)]):
    plt.subplot(2, 5, ii + 1)
    plt.imshow(image)
```

```
    plt.xticks([])
    plt.yticks([])
    plt.title(label)
plt.show()

plt.figure(figsize=(8, 3.5))
for (image, label), ii in zip(np_ds, [i for i in range(10)]):
    plt.subplot(2, 5, ii + 1)
    plt.imshow(image.mean(-1), cmap='gray')
    plt.xticks([])
    plt.yticks([])
    plt.title(label)
plt.show()
```

5 Data preprocessing

[67]:
```
def normalize_img(image, label):
    """Normalizes images: `uint8` -> `float32`."""
    return tf.cast(image, tf.float32) / 255., label

print(train_ds)
ds_train = train_ds.map(
    normalize_img, num_parallel_calls=tf.data.experimental.AUTOTUNE)
ds_train = ds_train.cache()
ds_train = ds_train.batch(128)
ds_train = ds_train.prefetch(tf.data.experimental.AUTOTUNE)
print(ds_train)
```

```
<DatasetV1Adapter shapes: ((32, 32, 3), ()), types: (tf.uint8, tf.int64)>
<DatasetV1Adapter shapes: ((None, 32, 32, 3), (None,)), types: (tf.float32,
tf.int64)>
```

[68]:
```
print(test_ds)
ds_test = test_ds.map(
    normalize_img, num_parallel_calls=tf.data.experimental.AUTOTUNE)
ds_test = ds_test.cache()
ds_test = ds_test.batch(128)
ds_test = ds_test.prefetch(tf.data.experimental.AUTOTUNE)
print(ds_test)
```

```
<DatasetV1Adapter shapes: ((32, 32, 3), ()), types: (tf.uint8, tf.int64)>
<DatasetV1Adapter shapes: ((None, 32, 32, 3), (None,)), types: (tf.float32,
tf.int64)>
```

6 Create callbacks

[69]:
```
checkpoint_every_epoch = ModelCheckpoint(cp_path_every_epoch +␣
↪"checkpoint_{epoch}_{loss}_{val_loss}.ckpt",
                                         save_best_only=False,␣
↪save_weights_only=True,
                                         mode='auto', save_freq='epoch')
checkpoint_best_epoch = ModelCheckpoint(cp_path_best_epoch + "checkpoint.ckpt",
                                         save_best_only=True,␣
↪save_weights_only=True,
                                         mode='auto', save_freq='epoch',␣
↪monitor="val_loss")
early_stopping = EarlyStopping(monitor='val_loss', min_delta=0, patience=5,
                               restore_best_weights=True)
```

```
reduce_on_plateau = ReduceLROnPlateau(monitor='val_loss', factor=0.1,␣
 ↪patience=3)
term_NaN = TerminateOnNaN()
# lr_shedul = LearningRateScheduler(lambda epoch: 1e-4 * 10**(1 / (epoch + 1)))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1)
```

[70]:
```
callbacks = [checkpoint_every_epoch, checkpoint_best_epoch, early_stopping,
             reduce_on_plateau, term_NaN, tensorboard_callback]
```

7 Create MLP model

[71]:
```
def create_MLP_model(img_shape):
    initializer = tf.keras.initializers.HeUniform()
    regularizer = tf.keras.regularizers.l2(1e-3)
    model = Sequential([
                        Flatten(input_shape=img_shape),
                        Dense(1024, activation='relu',
                              kernel_regularizer=regularizer,␣
 ↪kernel_initializer=initializer),
                        Dense(512, activation='relu',
                              kernel_regularizer=regularizer,␣
 ↪kernel_initializer=initializer),
                        Dense(256, activation='relu',
                              kernel_regularizer=regularizer,␣
 ↪kernel_initializer=initializer),
                        Dense(10, activation='softmax', name='Output_layer')
    ])
    model.summary()
    return model

model_mlp = create_MLP_model(img_shape)
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_12 (Flatten)         (None, 3072)              0
_________________________________________________________________
dense_22 (Dense)             (None, 1024)              3146752
_________________________________________________________________
dense_23 (Dense)             (None, 512)               524800
_________________________________________________________________
dense_24 (Dense)             (None, 256)               131328
_________________________________________________________________
Output_layer (Dense)         (None, 10)                2570
```

```
=================================================================
Total params: 3,805,450
Trainable params: 3,805,450
Non-trainable params: 0
_________________________________________________________________
```

8 Compile and fit MLP model

[72]:
```
model_mlp.compile(optimizer=tf.keras.optimizers.Adam(lr=0.0002),␣
 ↪loss='sparse_categorical_crossentropy', metrics=['acc'])
history_mlp = model_mlp.fit(ds_train, epochs=50,
                    validation_data=ds_test, verbose=1,
                    callbacks=callbacks)
history_mlp
```

```
Epoch 1/50
      2/Unknown - 0s 125ms/step - loss: 6.1990 - acc:
0.1289WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.0241s vs `on_train_batch_end` time: 0.2217s).
Check your callbacks.

WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the
batch time (batch time: 0.0241s vs `on_train_batch_end` time: 0.2217s). Check
your callbacks.

573/573 [==============================] - 24s 42ms/step - loss: 3.7932 - acc:
0.3320 - val_loss: 2.5923 - val_acc: 0.5119
Epoch 2/50
573/573 [==============================] - 4s 6ms/step - loss: 2.1135 - acc:
0.6092 - val_loss: 1.9413 - val_acc: 0.6204
Epoch 3/50
573/573 [==============================] - 3s 6ms/step - loss: 1.6873 - acc:
0.6763 - val_loss: 1.6817 - val_acc: 0.6607
Epoch 4/50
573/573 [==============================] - 4s 6ms/step - loss: 1.4943 - acc:
0.7082 - val_loss: 1.5095 - val_acc: 0.6978
Epoch 5/50
573/573 [==============================] - 3s 6ms/step - loss: 1.3753 - acc:
0.7293 - val_loss: 1.4175 - val_acc: 0.7108
Epoch 6/50
573/573 [==============================] - 4s 6ms/step - loss: 1.2951 - acc:
0.7419 - val_loss: 1.3519 - val_acc: 0.7232
Epoch 7/50
573/573 [==============================] - 3s 6ms/step - loss: 1.2298 - acc:
0.7536 - val_loss: 1.3061 - val_acc: 0.7294
Epoch 8/50
573/573 [==============================] - 3s 6ms/step - loss: 1.1751 - acc:
0.7631 - val_loss: 1.2537 - val_acc: 0.7394
```

```
Epoch 9/50
573/573 [==============================] - 3s 6ms/step - loss: 1.1306 - acc:
0.7702 - val_loss: 1.2263 - val_acc: 0.7415
Epoch 10/50
573/573 [==============================] - 3s 6ms/step - loss: 1.0908 - acc:
0.7765 - val_loss: 1.1936 - val_acc: 0.7465
Epoch 11/50
573/573 [==============================] - 3s 6ms/step - loss: 1.0562 - acc:
0.7831 - val_loss: 1.1490 - val_acc: 0.7563
Epoch 12/50
573/573 [==============================] - 3s 6ms/step - loss: 1.0252 - acc:
0.7873 - val_loss: 1.1260 - val_acc: 0.7599
Epoch 13/50
573/573 [==============================] - 3s 6ms/step - loss: 0.9951 - acc:
0.7927 - val_loss: 1.0939 - val_acc: 0.7673
Epoch 14/50
573/573 [==============================] - 4s 6ms/step - loss: 0.9689 - acc:
0.7990 - val_loss: 1.0727 - val_acc: 0.7724
Epoch 15/50
573/573 [==============================] - 4s 6ms/step - loss: 0.9454 - acc:
0.8028 - val_loss: 1.0578 - val_acc: 0.7752
Epoch 16/50
573/573 [==============================] - 4s 6ms/step - loss: 0.9211 - acc:
0.8088 - val_loss: 1.0234 - val_acc: 0.7846
Epoch 17/50
573/573 [==============================] - 3s 6ms/step - loss: 0.9023 - acc:
0.8115 - val_loss: 1.0126 - val_acc: 0.7842
Epoch 18/50
573/573 [==============================] - 3s 6ms/step - loss: 0.8874 - acc:
0.8143 - val_loss: 1.0164 - val_acc: 0.7812
Epoch 19/50
573/573 [==============================] - 3s 6ms/step - loss: 0.8728 - acc:
0.8169 - val_loss: 1.0082 - val_acc: 0.7821
Epoch 20/50
573/573 [==============================] - 3s 6ms/step - loss: 0.8582 - acc:
0.8193 - val_loss: 1.0003 - val_acc: 0.7830
Epoch 21/50
573/573 [==============================] - 3s 6ms/step - loss: 0.8426 - acc:
0.8230 - val_loss: 0.9763 - val_acc: 0.7907
Epoch 22/50
573/573 [==============================] - 3s 6ms/step - loss: 0.8308 - acc:
0.8259 - val_loss: 0.9807 - val_acc: 0.7875
Epoch 23/50
573/573 [==============================] - 3s 6ms/step - loss: 0.8199 - acc:
0.8277 - val_loss: 0.9435 - val_acc: 0.7994
Epoch 24/50
573/573 [==============================] - 3s 6ms/step - loss: 0.8117 - acc:
0.8291 - val_loss: 0.9456 - val_acc: 0.7976
```

```
Epoch 25/50
573/573 [==============================] - 3s 6ms/step - loss: 0.8021 - acc:
0.8310 - val_loss: 0.9224 - val_acc: 0.8053
Epoch 26/50
573/573 [==============================] - 3s 6ms/step - loss: 0.7884 - acc:
0.8347 - val_loss: 0.9288 - val_acc: 0.8009
Epoch 27/50
573/573 [==============================] - 3s 6ms/step - loss: 0.7819 - acc:
0.8366 - val_loss: 0.9348 - val_acc: 0.7981
Epoch 28/50
573/573 [==============================] - 3s 6ms/step - loss: 0.7733 - acc:
0.8387 - val_loss: 0.9373 - val_acc: 0.7964
Epoch 29/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6961 - acc:
0.8659 - val_loss: 0.8449 - val_acc: 0.8300
Epoch 30/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6868 - acc:
0.8691 - val_loss: 0.8419 - val_acc: 0.8307
Epoch 31/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6833 - acc:
0.8701 - val_loss: 0.8399 - val_acc: 0.8315
Epoch 32/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6802 - acc:
0.8707 - val_loss: 0.8369 - val_acc: 0.8316
Epoch 33/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6775 - acc:
0.8716 - val_loss: 0.8355 - val_acc: 0.8318
Epoch 34/50
573/573 [==============================] - 4s 6ms/step - loss: 0.6748 - acc:
0.8721 - val_loss: 0.8324 - val_acc: 0.8323
Epoch 35/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6721 - acc:
0.8728 - val_loss: 0.8307 - val_acc: 0.8326
Epoch 36/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6697 - acc:
0.8735 - val_loss: 0.8285 - val_acc: 0.8332
Epoch 37/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6673 - acc:
0.8739 - val_loss: 0.8268 - val_acc: 0.8334
Epoch 38/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6651 - acc:
0.8742 - val_loss: 0.8250 - val_acc: 0.8336
Epoch 39/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6628 - acc:
0.8748 - val_loss: 0.8235 - val_acc: 0.8337
Epoch 40/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6607 - acc:
0.8752 - val_loss: 0.8219 - val_acc: 0.8343
```

```
Epoch 41/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6586 - acc:
0.8755 - val_loss: 0.8206 - val_acc: 0.8347
Epoch 42/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6566 - acc:
0.8761 - val_loss: 0.8195 - val_acc: 0.8346
Epoch 43/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6547 - acc:
0.8766 - val_loss: 0.8181 - val_acc: 0.8347
Epoch 44/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6528 - acc:
0.8771 - val_loss: 0.8167 - val_acc: 0.8352
Epoch 45/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6509 - acc:
0.8775 - val_loss: 0.8153 - val_acc: 0.8355
Epoch 46/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6491 - acc:
0.8781 - val_loss: 0.8141 - val_acc: 0.8355
Epoch 47/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6473 - acc:
0.8784 - val_loss: 0.8131 - val_acc: 0.8358
Epoch 48/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6456 - acc:
0.8787 - val_loss: 0.8121 - val_acc: 0.8362
Epoch 49/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6439 - acc:
0.8791 - val_loss: 0.8113 - val_acc: 0.8365
Epoch 50/50
573/573 [==============================] - 3s 6ms/step - loss: 0.6422 - acc:
0.8797 - val_loss: 0.8101 - val_acc: 0.8367
```

[72]: `<tensorflow.python.keras.callbacks.History at 0x7faebfd748d0>`

9 Create CNN model

[73]:
```
def create_CNN_model(img_shape):
    initializer = tf.keras.initializers.HeUniform()
    # regularizer dont improve; Dropout better
    regularizer = None # tf.keras.regularizers.l2(2e-5)

    model = Sequential([
                        Conv2D(32, 3, input_shape=(32, 32, 3), padding='same',␣
↪activation='relu',
                               kernel_regularizer=regularizer,␣
↪kernel_initializer=initializer,
                               name='Input_layer'),
```

```
                        Conv2D(32, 3, padding='same', activation='relu',
                               kernel_regularizer=regularizer,␣
↪kernel_initializer=initializer),
                        BatchNormalization(),
                        MaxPool2D(2),
                        Dropout(0.4),
                        Conv2D(64, 3, padding='same', activation='relu',
                               kernel_regularizer=regularizer,␣
↪kernel_initializer=initializer),
                        Conv2D(64, 3, padding='same', activation='relu',
                               kernel_regularizer=regularizer,␣
↪kernel_initializer=initializer),
                        BatchNormalization(),
                        MaxPool2D(2),
                        Dropout(0.4),
                        Conv2D(128, 3, padding='same', activation='relu',
                               kernel_regularizer=regularizer,␣
↪kernel_initializer=initializer),
                        Conv2D(128, 3, padding='same', activation='relu',
                               kernel_regularizer=regularizer,␣
↪kernel_initializer=initializer),
                        BatchNormalization(),
                        MaxPool2D(2),
                        Dropout(0.4),
                        Flatten(),
                        #GlobalMaxPool2D(),
                        Dense(128, activation='relu',
                               kernel_regularizer=regularizer,␣
↪kernel_initializer=initializer),
                        BatchNormalization(),
                        Dropout(0.5),
                        Dense(10, activation='softmax', name='Output_layer')
    ])
    model.summary()
    return model

model = create_CNN_model(img_shape)
```

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input_layer (Conv2D)         (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_35 (Conv2D)           (None, 32, 32, 32)        9248
_________________________________________________________________
batch_normalization_28 (Batc (None, 32, 32, 32)        128
```

```
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
dropout_28 (Dropout)         (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
batch_normalization_29 (Batc (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
dropout_29 (Dropout)         (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 8, 8, 128)         147584
_________________________________________________________________
batch_normalization_30 (Batc (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 4, 4, 128)         0
_________________________________________________________________
dropout_30 (Dropout)         (None, 4, 4, 128)         0
_________________________________________________________________
flatten_13 (Flatten)         (None, 2048)              0
_________________________________________________________________
dense_25 (Dense)             (None, 128)               262272
_________________________________________________________________
batch_normalization_31 (Batc (None, 128)               512
_________________________________________________________________
dropout_31 (Dropout)         (None, 128)               0
_________________________________________________________________
Output_layer (Dense)         (None, 10)                1290
=================================================================
Total params: 551,978
Trainable params: 551,274
Non-trainable params: 704
_________________________________________________________________
```

10 Compile and fit CNN model

[74]:
```
# clear previous history
!rm -rf checkpoint_best_epoch checkpoint_every_epoch logs_fit

model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),␣
 ↪loss='sparse_categorical_crossentropy', metrics=['acc'])
```

```
history = model.fit(ds_train, epochs=50,
                    validation_data=ds_test, verbose=1,
                    callbacks=callbacks)
history
```

```
Epoch 1/50
      2/Unknown - 0s 60ms/step - loss: 3.5918 - acc:
0.1133WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 0.0139s vs `on_train_batch_end` time: 0.1068s).
Check your callbacks.

WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the
batch time (batch time: 0.0139s vs `on_train_batch_end` time: 0.1068s). Check
your callbacks.

573/573 [==============================] - 9s 16ms/step - loss: 1.8782 - acc:
0.3980 - val_loss: 0.7196 - val_acc: 0.7725
Epoch 2/50
573/573 [==============================] - 9s 15ms/step - loss: 0.6050 - acc:
0.8108 - val_loss: 0.3828 - val_acc: 0.8843
Epoch 3/50
573/573 [==============================] - 9s 15ms/step - loss: 0.4561 - acc:
0.8602 - val_loss: 0.3222 - val_acc: 0.9049
Epoch 4/50
573/573 [==============================] - 9s 15ms/step - loss: 0.3887 - acc:
0.8817 - val_loss: 0.2823 - val_acc: 0.9174
Epoch 5/50
573/573 [==============================] - 9s 15ms/step - loss: 0.3530 - acc:
0.8935 - val_loss: 0.2882 - val_acc: 0.9146
Epoch 6/50
573/573 [==============================] - 9s 15ms/step - loss: 0.3213 - acc:
0.9040 - val_loss: 0.2499 - val_acc: 0.9285
Epoch 7/50
573/573 [==============================] - 9s 15ms/step - loss: 0.3012 - acc:
0.9096 - val_loss: 0.2308 - val_acc: 0.9345
Epoch 8/50
573/573 [==============================] - 9s 15ms/step - loss: 0.2865 - acc:
0.9143 - val_loss: 0.2264 - val_acc: 0.9347
Epoch 9/50
573/573 [==============================] - 9s 15ms/step - loss: 0.2691 - acc:
0.9196 - val_loss: 0.2074 - val_acc: 0.9414
Epoch 10/50
573/573 [==============================] - 9s 15ms/step - loss: 0.2547 - acc:
0.9245 - val_loss: 0.2005 - val_acc: 0.9437
Epoch 11/50
573/573 [==============================] - 9s 15ms/step - loss: 0.2455 - acc:
0.9282 - val_loss: 0.1920 - val_acc: 0.9467
Epoch 12/50
```

```
573/573 [==============================] - 9s 15ms/step - loss: 0.2368 - acc:
0.9295 - val_loss: 0.1882 - val_acc: 0.9479
Epoch 13/50
573/573 [==============================] - 9s 15ms/step - loss: 0.2291 - acc:
0.9320 - val_loss: 0.1943 - val_acc: 0.9446
Epoch 14/50
573/573 [==============================] - 9s 15ms/step - loss: 0.2212 - acc:
0.9341 - val_loss: 0.1942 - val_acc: 0.9465
Epoch 15/50
573/573 [==============================] - 9s 15ms/step - loss: 0.2115 - acc:
0.9381 - val_loss: 0.1797 - val_acc: 0.9503
Epoch 16/50
573/573 [==============================] - 9s 15ms/step - loss: 0.2092 - acc:
0.9380 - val_loss: 0.1808 - val_acc: 0.9504
Epoch 17/50
573/573 [==============================] - 9s 15ms/step - loss: 0.2029 - acc:
0.9400 - val_loss: 0.1795 - val_acc: 0.9521
Epoch 18/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1932 - acc:
0.9426 - val_loss: 0.1759 - val_acc: 0.9534
Epoch 19/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1872 - acc:
0.9449 - val_loss: 0.1701 - val_acc: 0.9551
Epoch 20/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1819 - acc:
0.9465 - val_loss: 0.1686 - val_acc: 0.9558
Epoch 21/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1814 - acc:
0.9470 - val_loss: 0.1689 - val_acc: 0.9557
Epoch 22/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1754 - acc:
0.9488 - val_loss: 0.1689 - val_acc: 0.9561
Epoch 23/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1699 - acc:
0.9496 - val_loss: 0.1662 - val_acc: 0.9566
Epoch 24/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1652 - acc:
0.9509 - val_loss: 0.1698 - val_acc: 0.9552
Epoch 25/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1624 - acc:
0.9518 - val_loss: 0.1672 - val_acc: 0.9560
Epoch 26/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1610 - acc:
0.9532 - val_loss: 0.1764 - val_acc: 0.9538
Epoch 27/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1389 - acc:
0.9589 - val_loss: 0.1626 - val_acc: 0.9597
Epoch 28/50
```

```
573/573 [==============================] - 9s 15ms/step - loss: 0.1337 - acc:
0.9603 - val_loss: 0.1617 - val_acc: 0.9595
Epoch 29/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1279 - acc:
0.9621 - val_loss: 0.1614 - val_acc: 0.9589
Epoch 30/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1248 - acc:
0.9626 - val_loss: 0.1621 - val_acc: 0.9586
Epoch 31/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1184 - acc:
0.9651 - val_loss: 0.1608 - val_acc: 0.9599
Epoch 32/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1180 - acc:
0.9656 - val_loss: 0.1625 - val_acc: 0.9589
Epoch 33/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1202 - acc:
0.9641 - val_loss: 0.1602 - val_acc: 0.9597
Epoch 34/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1161 - acc:
0.9655 - val_loss: 0.1605 - val_acc: 0.9589
Epoch 35/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1150 - acc:
0.9661 - val_loss: 0.1594 - val_acc: 0.9601
Epoch 36/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1149 - acc:
0.9656 - val_loss: 0.1605 - val_acc: 0.9606
Epoch 37/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1116 - acc:
0.9665 - val_loss: 0.1610 - val_acc: 0.9599
Epoch 38/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1093 - acc:
0.9676 - val_loss: 0.1633 - val_acc: 0.9590
Epoch 39/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1067 - acc:
0.9689 - val_loss: 0.1600 - val_acc: 0.9609
Epoch 40/50
573/573 [==============================] - 9s 15ms/step - loss: 0.1077 - acc:
0.9678 - val_loss: 0.1598 - val_acc: 0.9609
```

[74]: <tensorflow.python.keras.callbacks.History at 0x7fae5ddd57b8>

11 Evaluate model

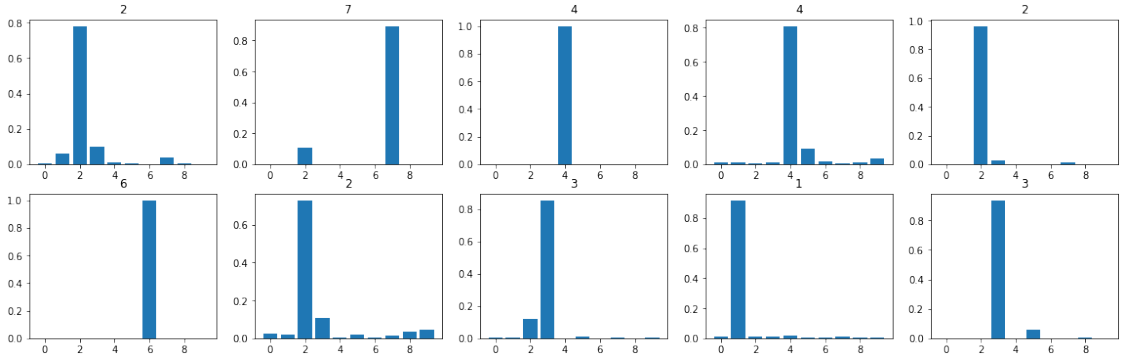
[75]:
```
model.evaluate(ds_test)
```

```
204/204 [==============================] - 1s 5ms/step - loss: 0.1594 - acc:
0.9601
```

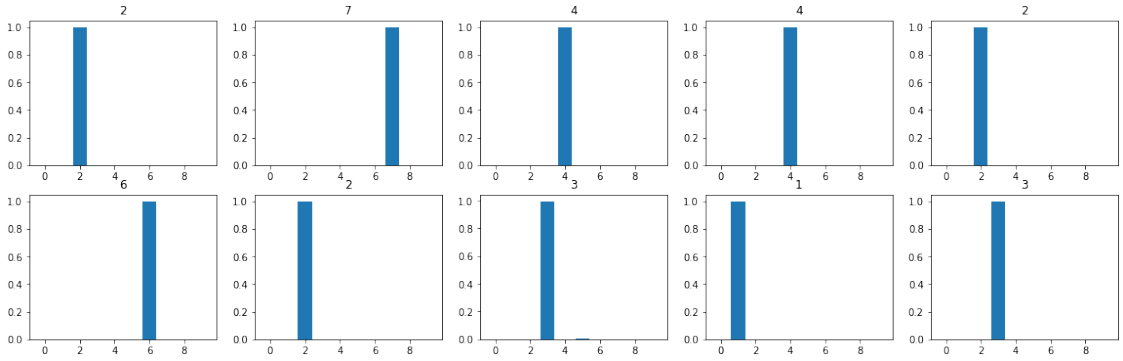
[75]: [0.15944360196590424, 0.9601259827613831]

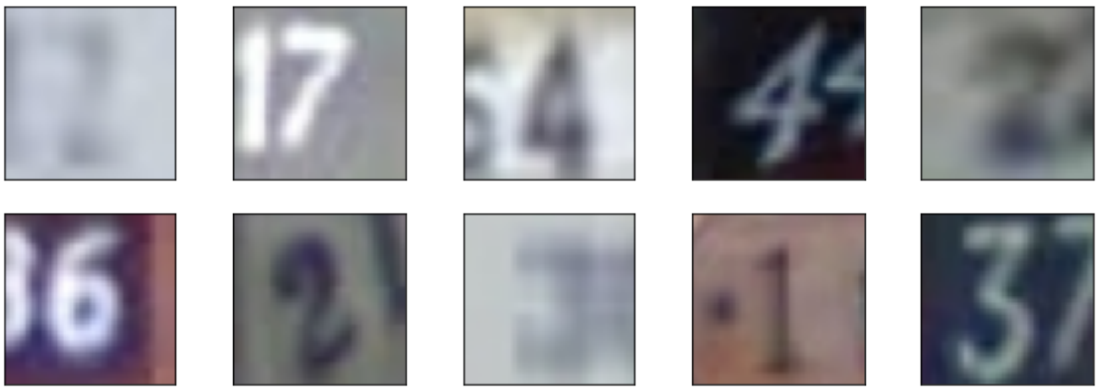
[81]:
```
predictions = model.predict(ds_test)[:10]

for images, labels in ds_test.take(1):
    labels = labels.numpy()
    images = images.numpy()
x = np.array([i for i in range(0, 10)])

print("MLP classifier")
predictions = model_mlp.predict(ds_test)[:10]
plt.figure(figsize=(20, 6))
for (ii, val) in enumerate(predictions):
    plt.subplot(2, 5, ii + 1)
    plt.bar(x, val)
    plt.title(labels[ii])
plt.show()

print("CNN classifier")
predictions = model.predict(ds_test)[:10]
plt.figure(figsize=(20, 6))
for (ii, val) in enumerate(predictions):
    plt.subplot(2, 5, ii + 1)
    plt.bar(x, val)
    plt.title(labels[ii])
plt.show()

plt.figure(figsize=(10, 3.5))
for (ii, image) in enumerate(images[:10]):
    plt.subplot(2, 5, ii + 1)
    plt.imshow(image)
    plt.xticks([])
    plt.yticks([])
plt.show()
```

```
MLP classifier
```



CNN classifier





12 Visualize history

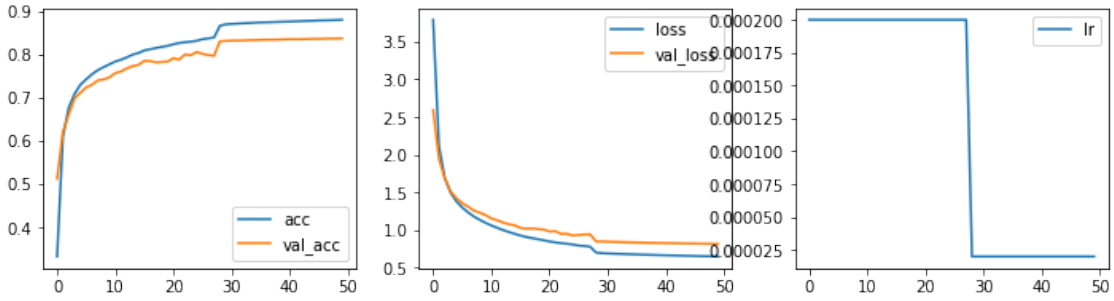
[77]:
```
%tensorboard --logdir logs_fit
```

<IPython.core.display.Javascript object>

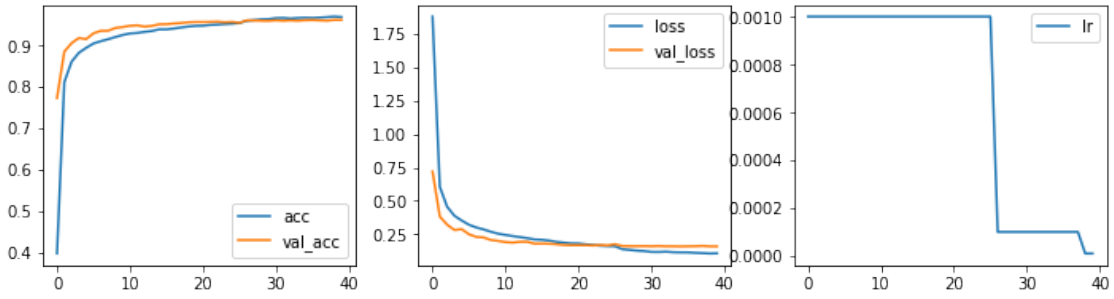
[78]:
```
def show_hostory(history):
    plt.figure(figsize=(12, 3))
    # print(history.history.keys())
    plt.subplot(1, 3, 1)
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.legend(['acc', 'val_acc'])
    plt.subplot(1, 3, 2)
```

```
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.legend(['loss', 'val_loss'])
    plt.subplot(1, 3, 3)
    plt.plot(history.history['lr'])
    plt.legend(['lr'])
    plt.show()

print('MLP history')
show_hostory(history_mlp)
print('CNN history')
show_hostory(history)
```

```
MLP history
```




```
CNN history
```



13 Load weights and save model

[79]:
```
model.load_weights(tf.train.latest_checkpoint(cp_path_best_epoch))
model.evaluate(ds_test)

model.save('final_model')
```

```
# keras model
model.save('final_tf_model.h5')
```

```
204/204 [==============================] - 1s 5ms/step - loss: 0.1594 - acc:
0.9601
INFO:tensorflow:Assets written to: final_model/assets

INFO:tensorflow:Assets written to: final_model/assets
```

[80]:
```
!ls
```

```
checkpoint_best_epoch   final_model        logs_fit
checkpoint_every_epoch  final_tf_model.h5  sample_data
```