Image classifier for the SVHN dataset

In [1]:
```
import tensorflow as tf
from scipy.io import loadmat

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

In [2]:
```
# Loading the dataset

train = loadmat('train_32x32.mat')
test = loadmat('test_32x32.mat')
```

In [3]:
```
# Inspecting the dataset

print(train.keys())
print(train['X'].shape)
print(np.unique(train['y']))
print(np.unique(test['y']))
```

```
dict_keys(['__header__', '__version__', '__globals__', 'X', 'y'])
(32, 32, 3, 73257)
[ 1  2  3  4  5  6  7  8  9 10]
[ 1  2  3  4  5  6  7  8  9 10]
```

In [4]:
```
#Divided images values by 255 so they fall between 0 and 1
train_labels = train['y']
train_images = train['X']/255
test_labels = test['y']
test_images = test['X']/255
```

In [5]:
```
#Labels that are the number 10 correspond to training images of a zero. The
#the output layer softmax in the model won't work

train_labels[train_labels[:]==10] = 0
test_labels[test_labels[:]==10] = 0
```

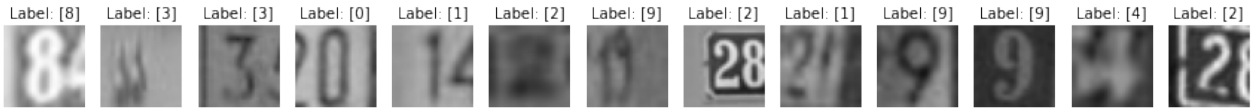
In [6]:
```
#Suffle images
def shuffle(images, labels, m):
    random_index = list(np.random.permutation(m))
    random_im = images[:,:,:,random_index]

    random_labels = labels[random_index]

    return random_im, random_labels

train_images, train_labels = shuffle(train_images, train_labels, train_imag
```

In [7]:

```
# Select random training images to display

train_examples_num = train_images.shape[3] #number of train examples
random_ind = np.random.choice(train_examples_num, 13) #generate 13 numbers
random_train_im = train_images[..., random_ind] #take 13 random images

random_train_lab = train_labels[random_ind] #take corresponding labels

fig, axes = plt.subplots(nrows=1, ncols=13, figsize=(15, 15)) #generate sub

for (i, ax) in enumerate(axes.flat): # flatten the generated axes
    img = random_train_im[:,:,:,i] #take random image i
    ax.set_axis_off()
    ax.imshow(np.squeeze(img))
    ax.set_title('Label: ' +str(random_train_lab[i]), fontsize=10)
```

In [8]:

```
#Convert images to gray

def convert_gray(data, labels, print_it):
    num_toprint=13
    m = data.shape[3] #number of examples
    data = np.average(data, axis = 2)

    if print_it:
        ones = np.ones((32, 32, 3, 1))

        random_ind = np.random.choice(m, num_toprint) #random index to prin
        random_data_gr = data[..., random_ind] #take images to print
        random_lab = labels[random_ind] #take labels

        print_data = random_data_gr[:,:,np.newaxis,:] * ones #add dummy inc
        print_labels = random_lab

        fig, axes = plt.subplots(nrows=1, ncols=num_toprint, figsize=(15, 1
        for (i, ax) in enumerate(axes.flat):

            img = print_data[:,:,:,i]
            ax.set_axis_off()
            ax.imshow(np.squeeze(img))
            ax.set_title('Label: ' +str(print_labels[i]), fontsize=10)

    return data.T

train_images = convert_gray(train_images,train_labels,True)
test_images_gr = convert_gray(test_images, test_labels, False)
```



MLP neural network classifier

In [9]:
```
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense, Flatten, Softmax, Conv2D, MaxPoo
from tensorflow.keras.callbacks import ModelCheckpoint, Callback, CSVLogger
import pandas as pd
```

In [10]:
```
def get_model(input_shape):

    model = Sequential([
            Flatten(input_shape=input_shape),
            Dense(128, activation='tanh', kernel_initializer='he_normal', k
                  kernel_regularizer=tf.keras.regularizers.l2(0.001)),
            Dense(64, activation='tanh',kernel_initializer='he_normal', bia
                  kernel_regularizer=tf.keras.regularizers.l2(0.001)),
            Dense(64, activation='tanh',kernel_initializer='he_normal', bia
                  kernel_regularizer=tf.keras.regularizers.l2(0.001)),

            Dense(10, activation='softmax')
    ])
    return model
```

In [11]:
```
print(train_images[0,:,:].shape)
```

```
(32, 32)
```

In [12]:
```
model = get_model(train_images[0,:,:].shape)
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 128)               131200
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 144,266
Trainable params: 144,266
Non-trainable params: 0
_________________________________________________________________
None
```

In [13]:

```
model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])
print(train_images[0,:,:].shape)
print(train_labels.shape)
print(train_images.shape)

checkpoint = ModelCheckpoint('Model_1', save_weights_only=False, save_best_

history = model.fit(train_images,train_labels, epochs=30, batch_size=10, va
                    callbacks=[checkpoint, tf.keras.callbacks.CSVLogger("Mo
```

```
(32, 32)
(73257, 1)
(73257, 32, 32)
Epoch 1/30
6227/6227 [==============================] - 10s 2ms/step - loss: 2.4780 -
accuracy: 0.2381 - val_loss: 2.1732 - val_accuracy: 0.3587
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 2/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.8800 - a
ccuracy: 0.4658 - val_loss: 1.7399 - val_accuracy: 0.5159
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 3/30
6227/6227 [==============================] - 9s 2ms/step - loss: 1.5959 - a
ccuracy: 0.5755 - val_loss: 1.5313 - val_accuracy: 0.6016
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 4/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.4627 - a
ccuracy: 0.6268 - val_loss: 1.4372 - val_accuracy: 0.6366
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 5/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.3783 - a
ccuracy: 0.6539 - val_loss: 1.3547 - val_accuracy: 0.6628
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 6/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.3139 - a
ccuracy: 0.6752 - val_loss: 1.3294 - val_accuracy: 0.6668
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 7/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.2690 - a
ccuracy: 0.6860 - val_loss: 1.2705 - val_accuracy: 0.6827
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 8/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.2293 - a
ccuracy: 0.6995 - val_loss: 1.2133 - val_accuracy: 0.7030
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 9/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.1962 - a
ccuracy: 0.7089 - val_loss: 1.2029 - val_accuracy: 0.7022
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 10/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.1687 - a
ccuracy: 0.7160 - val_loss: 1.1865 - val_accuracy: 0.7018
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 11/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.1466 - a
ccuracy: 0.7212 - val_loss: 1.1750 - val_accuracy: 0.7062
```

```
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 12/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.1270 - a
ccuracy: 0.7272 - val_loss: 1.1579 - val_accuracy: 0.7108
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 13/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.1092 - a
ccuracy: 0.7308 - val_loss: 1.1376 - val_accuracy: 0.7210
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 14/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.0919 - a
ccuracy: 0.7363 - val_loss: 1.1155 - val_accuracy: 0.7294
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 15/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.0795 - a
ccuracy: 0.7387 - val_loss: 1.0809 - val_accuracy: 0.7371
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 16/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.0638 - a
ccuracy: 0.7421 - val_loss: 1.1114 - val_accuracy: 0.7265
Epoch 17/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.0498 - a
ccuracy: 0.7489 - val_loss: 1.0616 - val_accuracy: 0.7431
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 18/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.0418 - a
ccuracy: 0.7489 - val_loss: 1.0930 - val_accuracy: 0.7303
Epoch 19/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.0309 - a
ccuracy: 0.7512 - val_loss: 1.1088 - val_accuracy: 0.7257
Epoch 20/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.0212 - a
ccuracy: 0.7544 - val_loss: 1.0407 - val_accuracy: 0.7460
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 21/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.0131 - a
ccuracy: 0.7571 - val_loss: 1.0769 - val_accuracy: 0.7372
Epoch 22/30
6227/6227 [==============================] - 9s 1ms/step - loss: 1.0024 - a
ccuracy: 0.7609 - val_loss: 1.0115 - val_accuracy: 0.7604
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 23/30
6227/6227 [==============================] - 9s 1ms/step - loss: 0.9946 - a
ccuracy: 0.7637 - val_loss: 1.0201 - val_accuracy: 0.7531
Epoch 24/30
6227/6227 [==============================] - 9s 1ms/step - loss: 0.9871 - a
ccuracy: 0.7658 - val_loss: 1.0356 - val_accuracy: 0.7478
Epoch 25/30
6227/6227 [==============================] - 9s 2ms/step - loss: 0.9787 - a
ccuracy: 0.7673 - val_loss: 1.0219 - val_accuracy: 0.7535
Epoch 26/30
6227/6227 [==============================] - 10s 2ms/step - loss: 0.9729 -
accuracy: 0.7697 - val_loss: 1.0031 - val_accuracy: 0.7580
INFO:tensorflow:Assets written to: Model_1/assets
Epoch 27/30
6227/6227 [==============================] - 9s 1ms/step - loss: 0.9652 - a
ccuracy: 0.7709 - val_loss: 1.0199 - val_accuracy: 0.7542
Epoch 28/30
6227/6227 [==============================] - 9s 1ms/step - loss: 0.9572 - a
ccuracy: 0.7746 - val_loss: 0.9915 - val_accuracy: 0.7643
INFO:tensorflow:Assets written to: Model_1/assets
```

```
Epoch 29/30
6227/6227 [==============================] - 9s 1ms/step - loss: 0.9522 - a
ccuracy: 0.7754 - val_loss: 1.0023 - val_accuracy: 0.7576
Epoch 30/30
6227/6227 [==============================] - 9s 1ms/step - loss: 0.9473 - a
ccuracy: 0.7783 - val_loss: 0.9778 - val_accuracy: 0.7640
INFO:tensorflow:Assets written to: Model_1/assets
```

In [14]:

```
pd.read_csv("Model1_results.csv", index_col = 'epoch')
```

Out[14]:

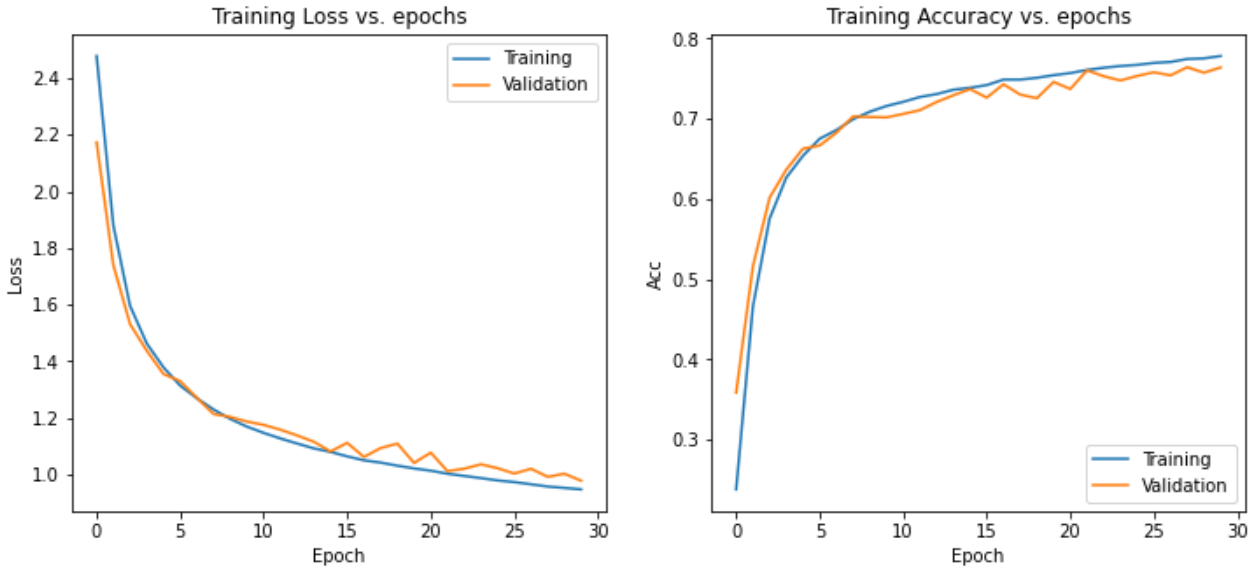
epoch	accuracy	loss	val_accuracy	val_loss
0	0.238084	2.478049	0.358722	2.173184
1	0.465777	1.879994	0.515880	1.739900
2	0.575512	1.595883	0.601602	1.531260
3	0.626775	1.462720	0.636637	1.437193
4	0.653899	1.378280	0.662754	1.354712
5	0.675194	1.313904	0.666758	1.329382
6	0.686002	1.268955	0.682683	1.270512
7	0.699525	1.229328	0.702976	1.213277
8	0.708887	1.196150	0.702248	1.202934
9	0.716018	1.168678	0.701793	1.186524
10	0.721173	1.146621	0.706161	1.175021
11	0.727244	1.126974	0.710802	1.157863
12	0.730841	1.109177	0.720994	1.137612
13	0.736253	1.091938	0.729366	1.115517
14	0.738710	1.079546	0.737101	1.080853
15	0.742083	1.063757	0.726454	1.111445
16	0.748940	1.049841	0.743107	1.061603
17	0.748908	1.041760	0.730276	1.092971
18	0.751172	1.030892	0.725726	1.108838
19	0.754416	1.021158	0.746019	1.040681
20	0.757066	1.013108	0.737192	1.076912
21	0.760937	1.002429	0.760397	1.011521
22	0.763715	0.994634	0.753117	1.020146
23	0.765819	0.987059	0.747839	1.035562
24	0.767312	0.978705	0.753481	1.021911
25	0.769721	0.972923	0.758031	1.003064
26	0.770942	0.965171	0.754209	1.019864
27	0.774587	0.957152	0.764310	0.991527
28	0.775390	0.952242	0.757576	1.002277
29	0.778345	0.947269	0.764037	0.977804

In [15]:

```
fig = plt.figure(figsize=(12, 5))
fig.add_subplot(121)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Training Loss vs. epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='upper right')

fig.add_subplot(122)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Training Accuracy vs. epochs')
plt.ylabel('Acc')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='lower right')
```

Out[15]:

```
<matplotlib.legend.Legend at 0x7fabee302590>
```



In [16]:

```
#Loss and accuracy on test set

test = model.evaluate(test_images_gr, test_labels, batch_size = 10, return_
print("Loss",test['loss'])
print("Accuracy", test['accuracy'])
```

```
2604/2604 [==============================] - 2s 829us/step - loss: 1.0516 -
accuracy: 0.7488
Loss 1.051615834236145
Accuracy 0.7487707734107971
```

CNN neural network classifier

In [17]:

```
def CNN_model():
    model = Sequential([
                Conv2D(16,(3,3), padding='SAME', activation='relu', input_s
                BatchNormalization(),
                Dropout(.2),
                MaxPooling2D((3,3)),
                Conv2D(16,(3,3), padding='SAME', activation='relu'),
                BatchNormalization(),
                MaxPooling2D((3,3)),
                Flatten(),
                Dense(64, activation='relu',kernel_initializer='he_normal',
                  kernel_regularizer=tf.keras.regularizers.l2(0.001)),
                Dense(10, activation='softmax')

    ])
    return model
```

In [18]:

```
print(train_images[:,:,:].shape)
model2 = CNN_model()
```

```
(73257, 32, 32)
```

In [19]:

```
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 16)        160
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 16)        64
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 16)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 10, 10, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 16)        2320
_________________________________________________________________
batch_normalization_1 (Batch (None, 10, 10, 16)        64
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 144)               0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                9280
_________________________________________________________________
dense_5 (Dense)              (None, 10)                650
=================================================================
Total params: 12,538
Trainable params: 12,474
Non-trainable params: 64
_________________________________________________________________
```

In [20]:

```
model2.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])
print(train_images[0,:,:].shape)
print(train_labels.shape)
print(train_images.shape)

checkpoint = ModelCheckpoint('Model_2', save_weights_only=False, save_best_

history2 = model2.fit(train_images[...,np.newaxis],train_labels, epochs=30,
                    callbacks=[checkpoint, tf.keras.callbacks.CSVLogger("Mo
```

```
(32, 32)
(73257, 1)
(73257, 32, 32)
Epoch 1/30
6227/6227 [==============================] - 50s 8ms/step - loss: 1.9614 -
accuracy: 0.3953 - val_loss: 1.3333 - val_accuracy: 0.6132
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 2/30
6227/6227 [==============================] - 49s 8ms/step - loss: 1.0951 -
accuracy: 0.6896 - val_loss: 1.0029 - val_accuracy: 0.7195
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 3/30
6227/6227 [==============================] - 50s 8ms/step - loss: 0.8861 -
accuracy: 0.7607 - val_loss: 0.8737 - val_accuracy: 0.7675
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 4/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.7951 -
accuracy: 0.7877 - val_loss: 0.8059 - val_accuracy: 0.7904
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 5/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.7374 -
accuracy: 0.8050 - val_loss: 0.7717 - val_accuracy: 0.7980
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 6/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.6962 -
accuracy: 0.8153 - val_loss: 0.7275 - val_accuracy: 0.8055
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 7/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.6633 -
accuracy: 0.8261 - val_loss: 0.6842 - val_accuracy: 0.8242
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 8/30
6227/6227 [==============================] - 50s 8ms/step - loss: 0.6357 -
accuracy: 0.8336 - val_loss: 0.6771 - val_accuracy: 0.8206
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 9/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.6124 -
accuracy: 0.8398 - val_loss: 0.6440 - val_accuracy: 0.8298
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 10/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.5937 -
accuracy: 0.8426 - val_loss: 0.6336 - val_accuracy: 0.8367
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 11/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.5744 -
accuracy: 0.8490 - val_loss: 0.6120 - val_accuracy: 0.8378
```

```
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 12/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.5574 -
accuracy: 0.8535 - val_loss: 0.6215 - val_accuracy: 0.8340
Epoch 13/30
6227/6227 [==============================] - 48s 8ms/step - loss: 0.5407 -
accuracy: 0.8578 - val_loss: 0.6209 - val_accuracy: 0.8317
Epoch 14/30
6227/6227 [==============================] - 50s 8ms/step - loss: 0.5305 -
accuracy: 0.8600 - val_loss: 0.5871 - val_accuracy: 0.8440
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 15/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.5185 -
accuracy: 0.8633 - val_loss: 0.5701 - val_accuracy: 0.8489
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 16/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.5060 -
accuracy: 0.8655 - val_loss: 0.5630 - val_accuracy: 0.8481
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 17/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.4932 -
accuracy: 0.8693 - val_loss: 0.5613 - val_accuracy: 0.8498
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 18/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.4851 -
accuracy: 0.8717 - val_loss: 0.5370 - val_accuracy: 0.8590
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 19/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.4741 -
accuracy: 0.8743 - val_loss: 0.5447 - val_accuracy: 0.8537
Epoch 20/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.4651 -
accuracy: 0.8752 - val_loss: 0.5360 - val_accuracy: 0.8559
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 21/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.4569 -
accuracy: 0.8789 - val_loss: 0.5324 - val_accuracy: 0.8564
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 22/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.4483 -
accuracy: 0.8794 - val_loss: 0.5241 - val_accuracy: 0.8599
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 23/30
6227/6227 [==============================] - 48s 8ms/step - loss: 0.4413 -
accuracy: 0.8828 - val_loss: 0.5130 - val_accuracy: 0.8601
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 24/30
6227/6227 [==============================] - 48s 8ms/step - loss: 0.4350 -
accuracy: 0.8832 - val_loss: 0.5110 - val_accuracy: 0.8614
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 25/30
6227/6227 [==============================] - 48s 8ms/step - loss: 0.4317 -
accuracy: 0.8830 - val_loss: 0.5109 - val_accuracy: 0.8600
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 26/30
6227/6227 [==============================] - 48s 8ms/step - loss: 0.4241 -
accuracy: 0.8857 - val_loss: 0.4922 - val_accuracy: 0.8671
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 27/30
6227/6227 [==============================] - 48s 8ms/step - loss: 0.4213 -
accuracy: 0.8865 - val_loss: 0.4914 - val_accuracy: 0.8698
```

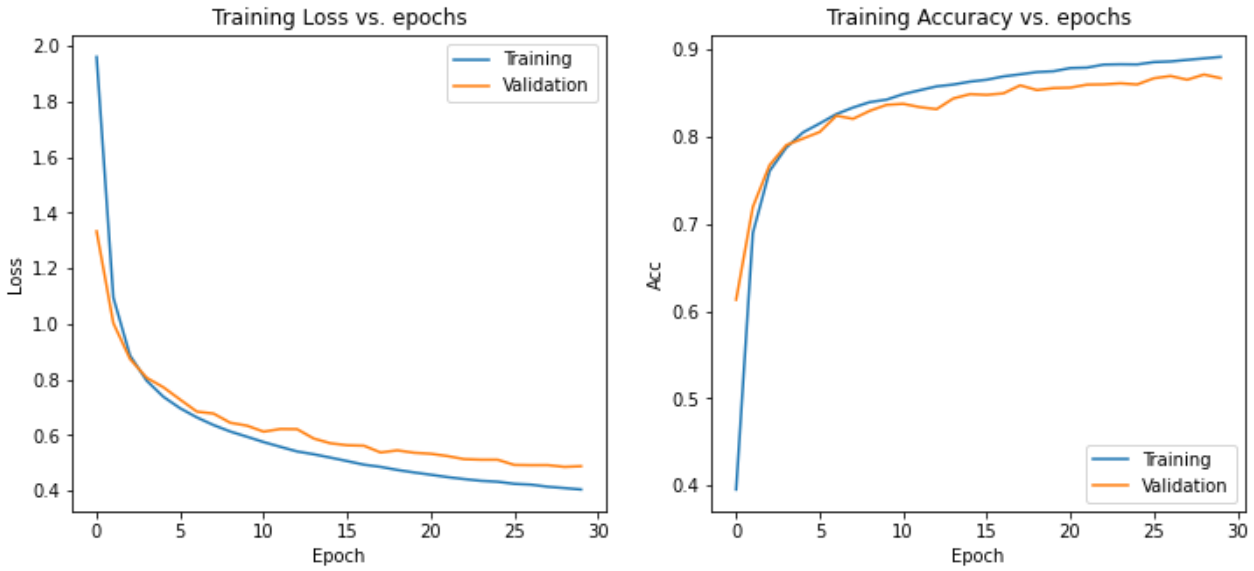
```
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 28/30
6227/6227 [==============================] - 48s 8ms/step - loss: 0.4136 -
accuracy: 0.8883 - val_loss: 0.4915 - val_accuracy: 0.8655
Epoch 29/30
6227/6227 [==============================] - 48s 8ms/step - loss: 0.4088 -
accuracy: 0.8900 - val_loss: 0.4851 - val_accuracy: 0.8714
INFO:tensorflow:Assets written to: Model_2/assets
Epoch 30/30
6227/6227 [==============================] - 49s 8ms/step - loss: 0.4037 -
accuracy: 0.8917 - val_loss: 0.4875 - val_accuracy: 0.8673
```

In [21]:

```
fig = plt.figure(figsize=(12, 5))
fig.add_subplot(121)
plt.plot(history2.history['loss'])
plt.plot(history2.history['val_loss'])
plt.title('Training Loss vs. epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='upper right')

fig.add_subplot(122)
plt.plot(history2.history['accuracy'])
plt.plot(history2.history['val_accuracy'])
plt.title('Training Accuracy vs. epochs')
plt.ylabel('Acc')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='lower right')
```

Out[21]:

```
<matplotlib.legend.Legend at 0x7fabed75f610>
```



In [22]:

```
#Loss and accuracy on test set

test = model2.evaluate(test_images_gr[...,np.newaxis], test_labels, batch_s
print("Loss",test['loss'])
print("Accuracy", test['accuracy'])
```

```
2604/2604 [==============================] - 6s 2ms/step - loss: 0.5318 - a
ccuracy: 0.8535
Loss 0.5318433046340942
Accuracy 0.8535264134407043
```

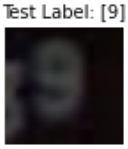
Models predictions

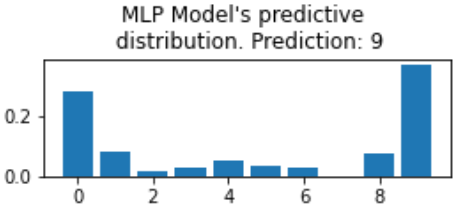
In [23]:
```
#Loading best weights and models for each version

MLP = load_model('Model_1')
CNN = load_model('Model_2')
```

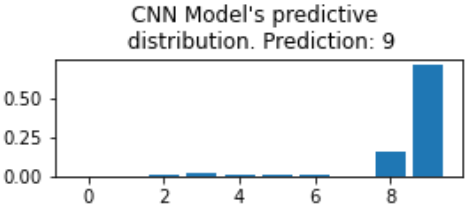
In [24]:
```
#Randomly select 5 images and labels from the test set

random_ix = np.random.choice(test_images.shape[3], 5) #random index to pri
random_im = test_images[..., random_ix] #take images to print
random_labels = test_labels[random_ix] #take labels
random_images_gr = convert_gray(random_im, random_labels, False)
```

In [45]:
```
#Results

predictionsMLP = MLP.predict(random_images_gr)
predictionsCNN = CNN.predict(random_images_gr[...,np.newaxis])

fig, axes = plt.subplots(5,3, figsize=(14, 10))
fig.subplots_adjust(hspace=.8, wspace=0.2)
for (i, ax) in enumerate(axes):
    axes[i, 0].set_axis_off()
    axes[i, 0].imshow(np.squeeze(random_im[:,:,:,i]))
    axes[i, 0].set_title('Test Label: ' +str(random_labels[i]), fontsize=1(
    axes[i, 1].bar(np.arange(len(predictionsMLP[i])), predictionsMLP[i])
    axes[i, 1].set_title(f"MLP Model's predictive \n distribution. Predicti
    axes[i, 2].bar(np.arange(len(predictionsCNN[i])), predictionsCNN[i])
    axes[i, 2].set_title(f"CNN Model's predictive \n distribution. Predicti
```

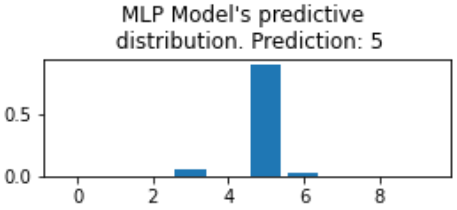
Test Label: [9]

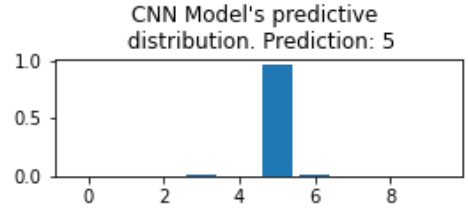




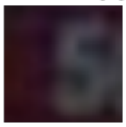


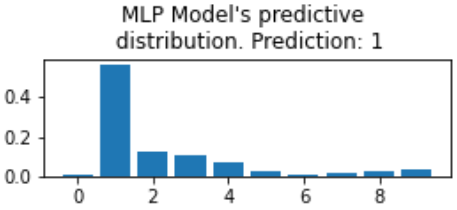
Test Label: [5]

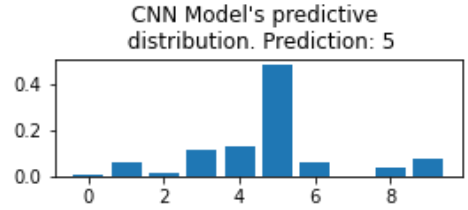




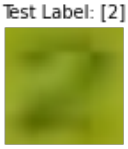
Test Label: [5]

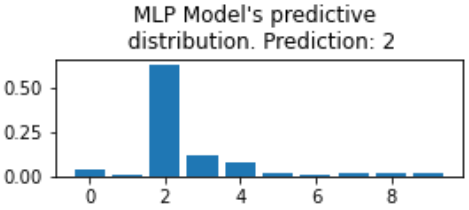


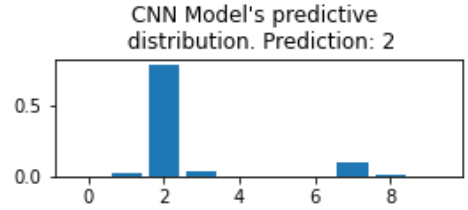




Test Label: [2]







Test Label: [9]