Capstone Project

July 7, 2020

1 Capstone Project

1.1 Image classifier for the SVHN dataset

1.1.1 Instructions

In this notebook, you will create a neural network that classifies real-world images digits. You will use concepts from throughout this course in building, training, testing, validating and saving your Tensorflow classifier model.

This project is peer-assessed. Within this notebook you will find instructions in each section for how to complete the project. Pay close attention to the instructions as the peer review will be carried out according to a grading rubric that checks key parts of the project instructions. Feel free to add extra cells into the notebook as required.

1.1.2 How to submit

When you have completed the Capstone project notebook, you will submit a pdf of the notebook for peer review. First ensure that the notebook has been fully executed from beginning to end, and all of the cell outputs are visible. This is important, as the grading rubric depends on the reviewer being able to view the outputs of your notebook. Save the notebook as a pdf (File -> Download as -> PDF via LaTeX). You should then submit this pdf for review.

1.1.3 Let's get started!

We'll start by running some imports, and loading the dataset. For this project you are free to make further imports throughout the notebook as you wish.

```
In [2]: import tensorflow as tf
        from scipy.io import loadmat
```

```
In [3]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Conv2D, Flatten, BatchNormalization, MaxPool2D, Der
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.model_selection import train_test_split
        %matplotlib inline
```

For the capstone project, you will use the SVHN dataset. This is an image dataset of over 600,000 digit images in all, and is a harder dataset than MNIST as the numbers appear in the context of natural scene images. SVHN is obtained from house numbers in Google Street View images.

- Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu and A. Y. Ng. "Reading Digits in Natural Images with Unsupervised Feature Learning". NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.

Your goal is to develop an end-to-end workflow for building, training, validating, evaluating and saving a neural network that classifies a real-world image into one of ten classes.

In [4]:
```
# Run this cell to load the dataset

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

Both `train` and `test` are dictionaries with keys `X` and `y` for the input images and labels respectively.

1.2 1. Inspect and preprocess the dataset

- Extract the training and testing images and labels separately from the train and test dictionaries loaded for you.
- Select a random sample of images and corresponding labels from the dataset (at least 10), and display them in a figure.
- Convert the training and test images to grayscale by taking the average across all colour channels for each pixel. *Hint: retain the channel dimension, which will now have size 1.*
- Select a random sample of the grayscale images and corresponding labels from the dataset (at least 10), and display them in a figure.

In [5]:
```
X_train = train['X']
X_test = test['X']
y_train = train['y']
y_test = test['y']
```

```
In [6]: X_train.shape, X_test.shape
```

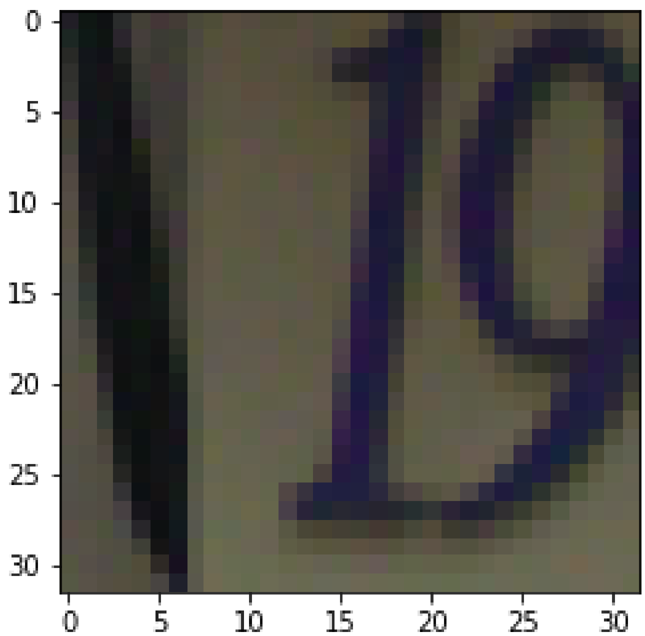
```
Out[6]: ((32, 32, 3, 73257), (32, 32, 3, 26032))
```

```
In [7]: X_train = np.moveaxis(X_train, -1, 0)
        X_test = np.moveaxis(X_test, -1 , 0)
```

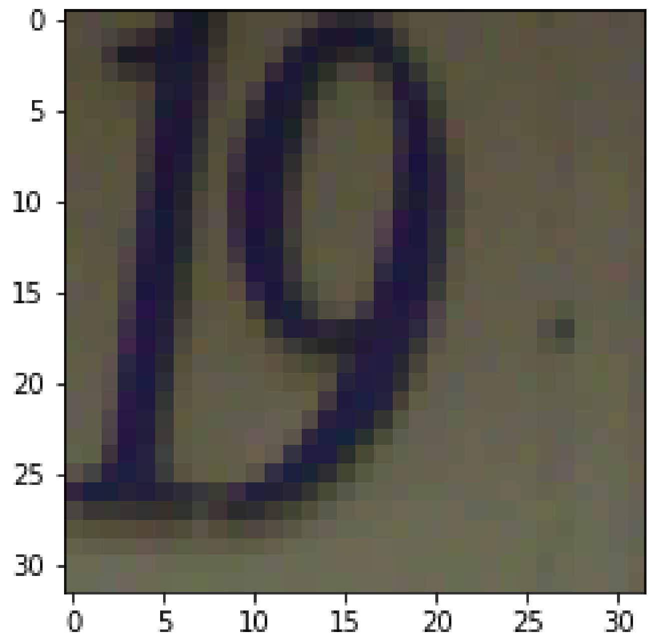
```
In [8]: X_train.shape, X_test.shape
```

```
Out[8]: ((73257, 32, 32, 3), (26032, 32, 32, 3))
```

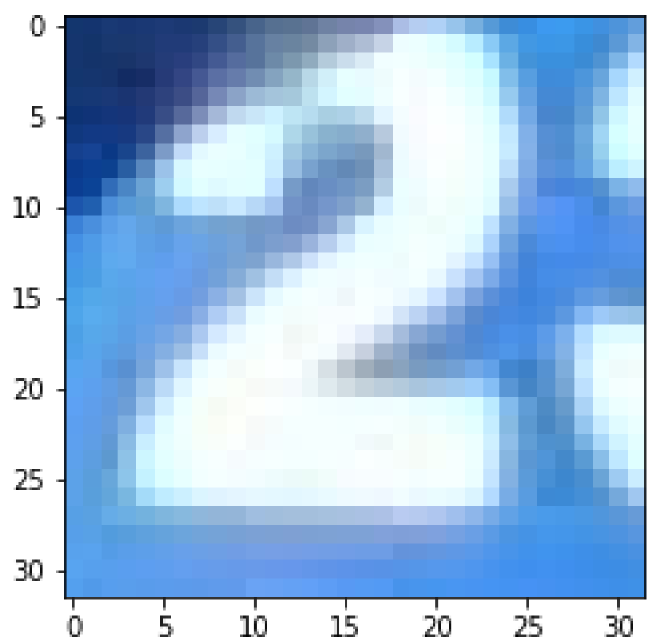
```
In [9]: for i in range(10):
            plt.imshow(X_train[i, :, :, :,])
            plt.show()
            print(y_train[i])
```



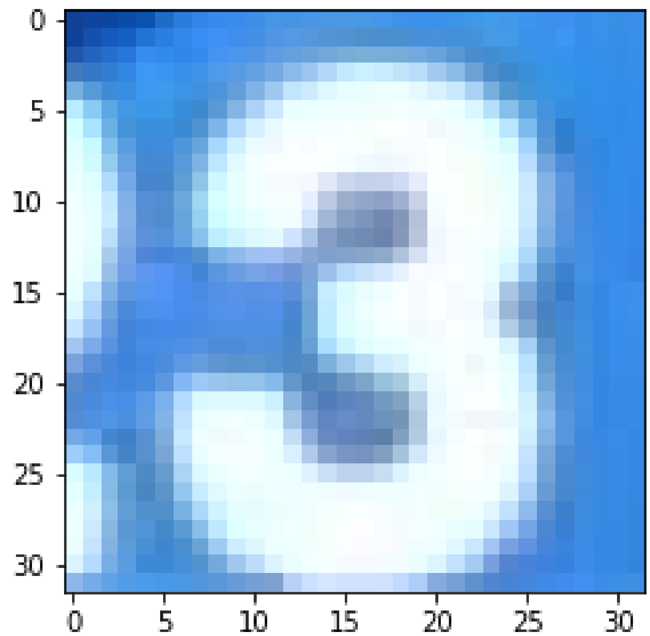

```
[1]
```



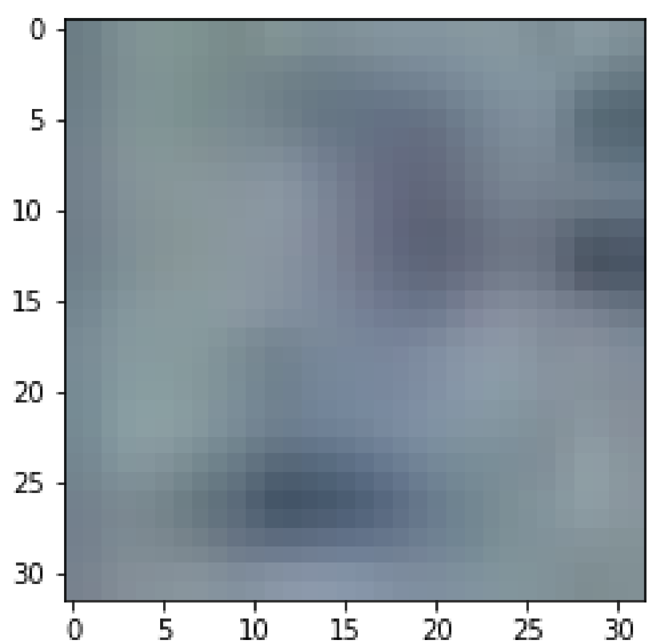
[9]



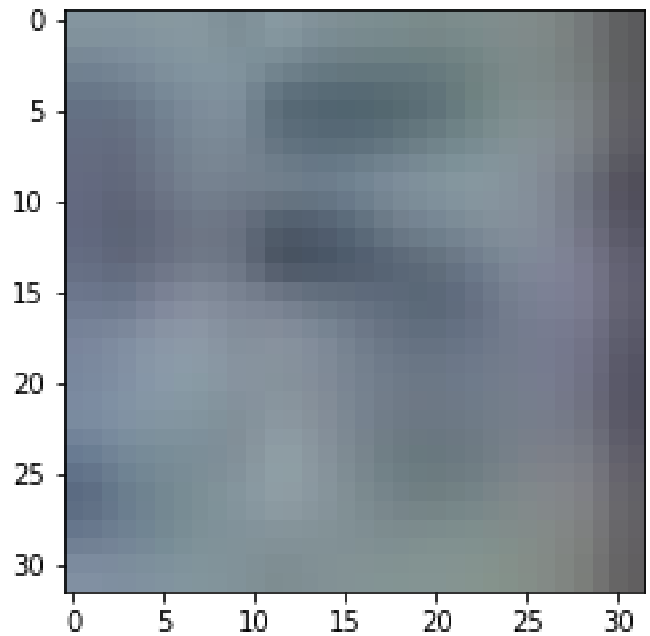
[2]



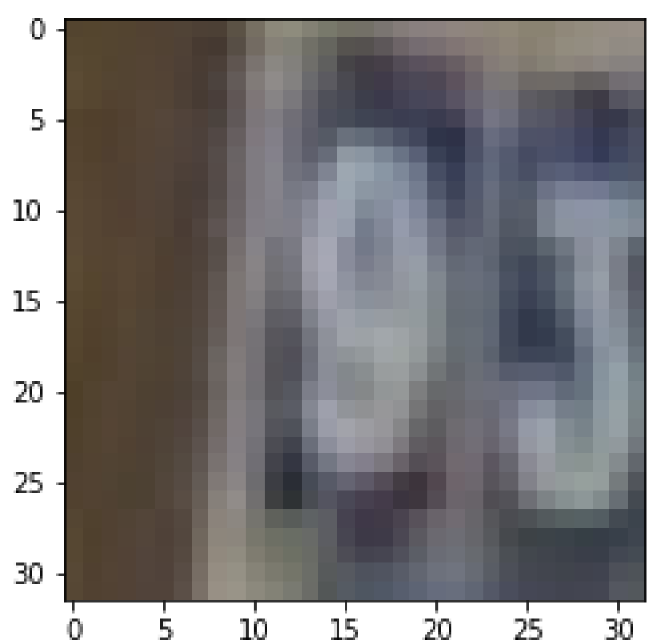
[3]



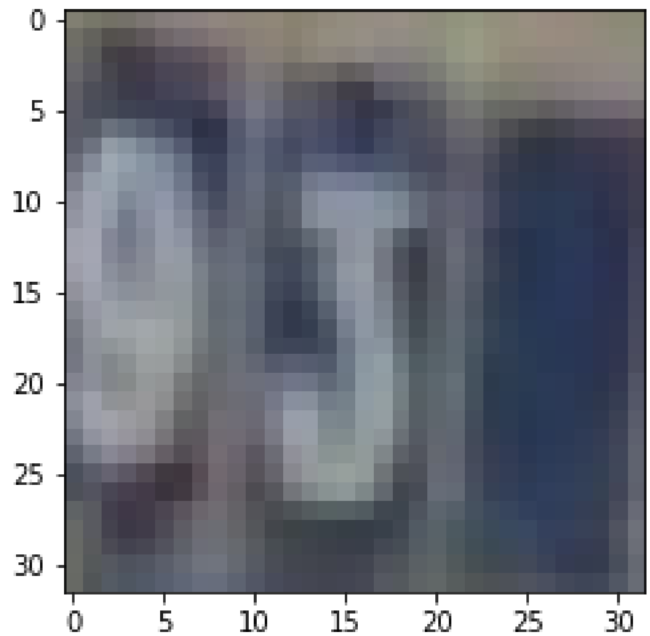
[2]



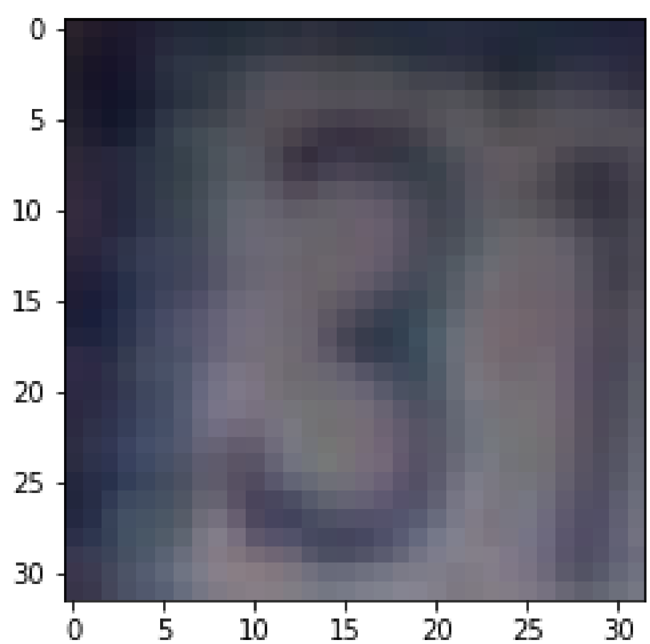
[5]



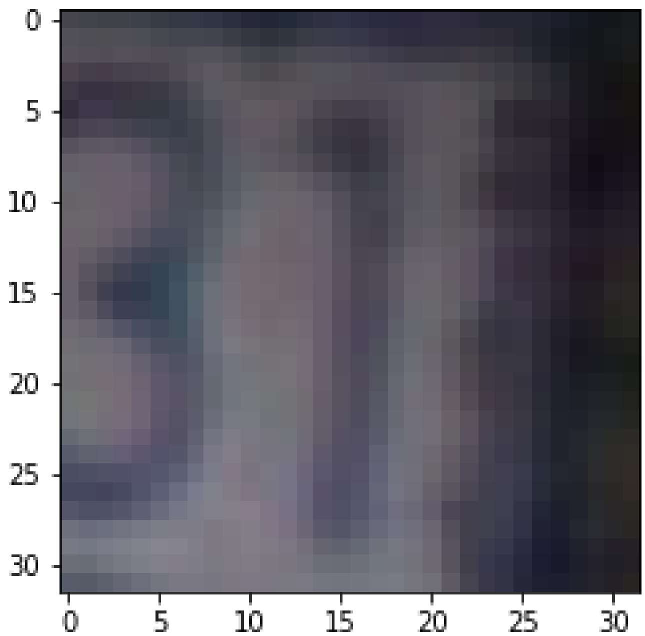
[9]



[3]



[3]



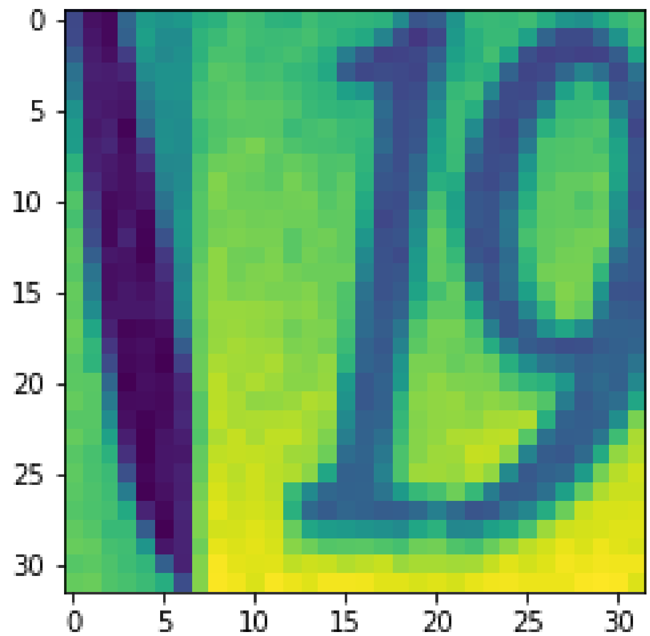
[1]

```
In [10]: X_train_gs = np.mean(X_train, 3).reshape(73257, 32, 32, 1)/255
         X_test_gs = np.mean(X_test,3).reshape(26032, 32,32 ,1)/255
         X_train_for_plotting = np.mean(X_train,3)
```

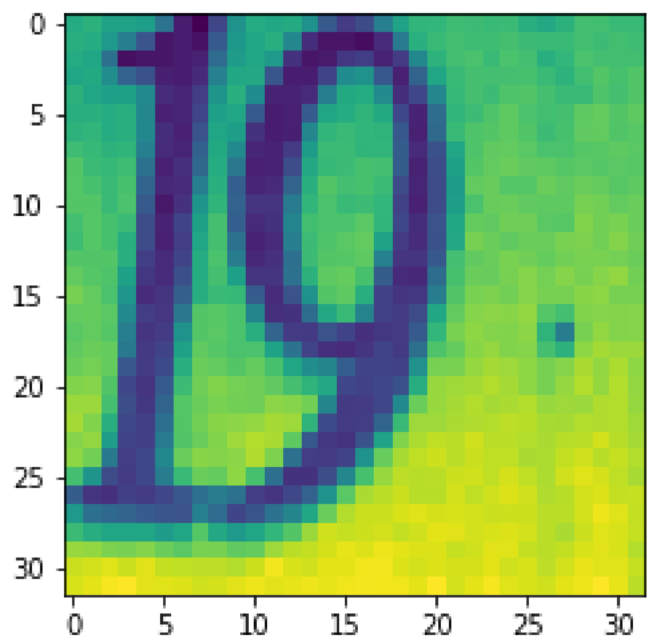
```
In [11]: X_train_gs.shape
```

```
Out[11]: (73257, 32, 32, 1)
```

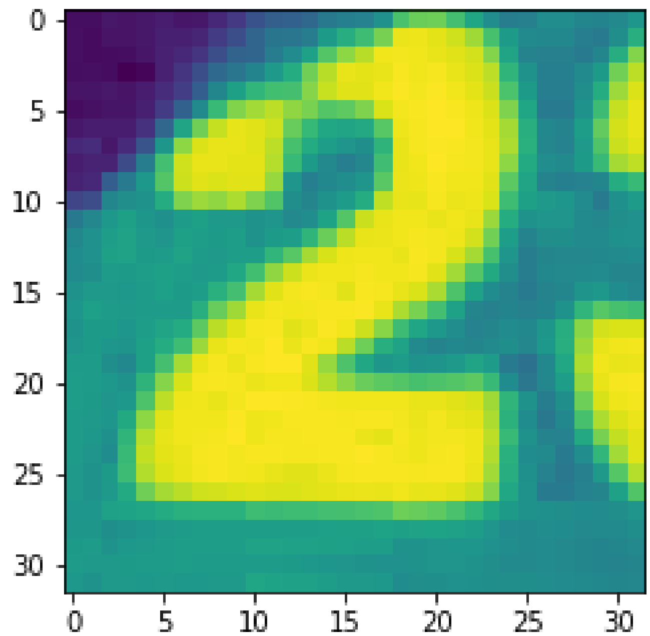
```
In [12]: for i in range(10):
             plt.imshow(X_train_for_plotting[i, :, :,])
             plt.show()
             print(y_train[i])
```



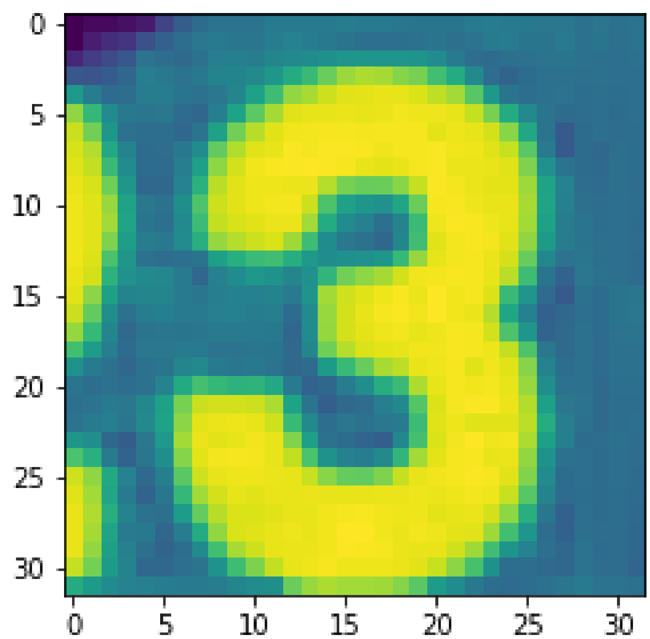
[1]



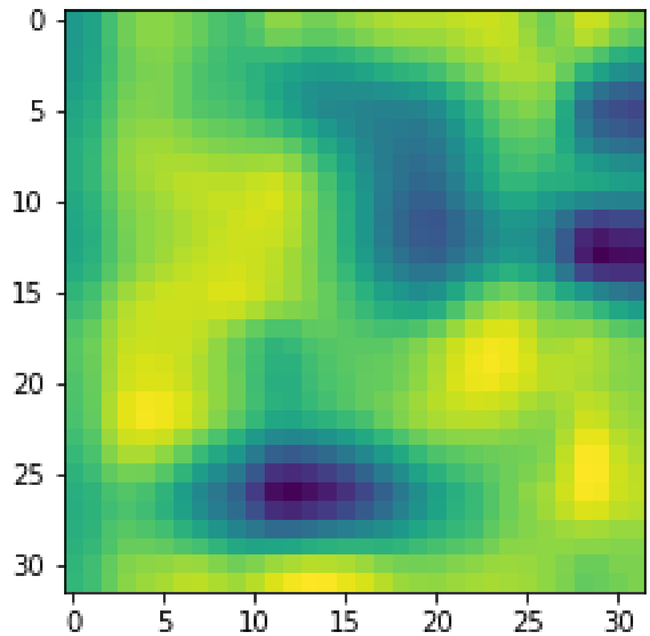
[9]



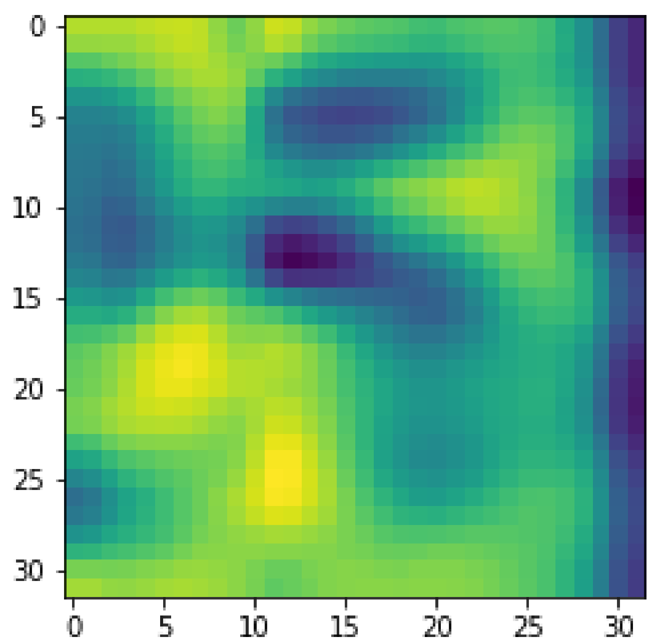
[2]



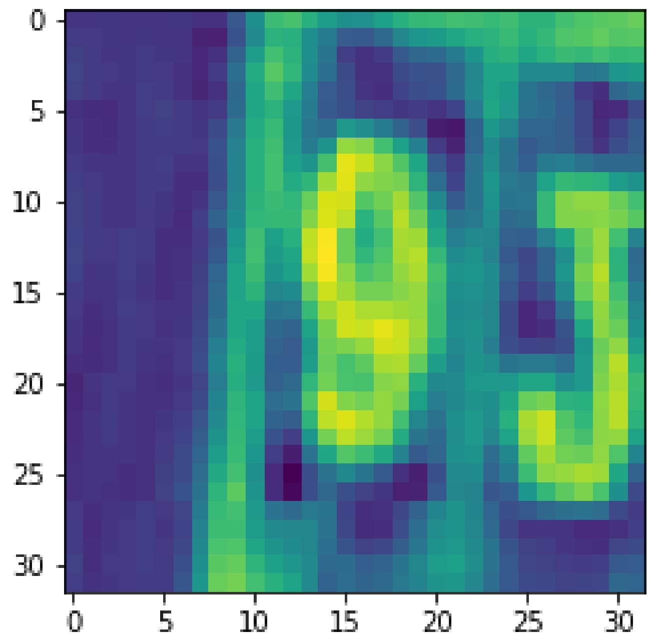
[3]



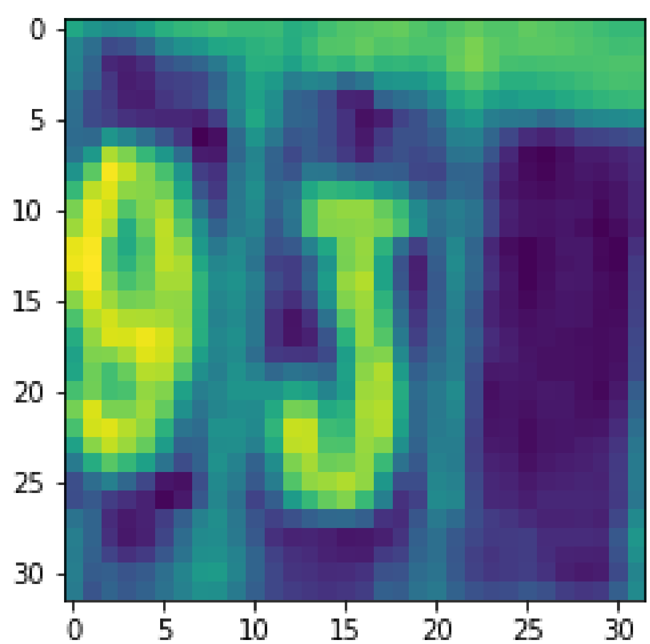
[2]



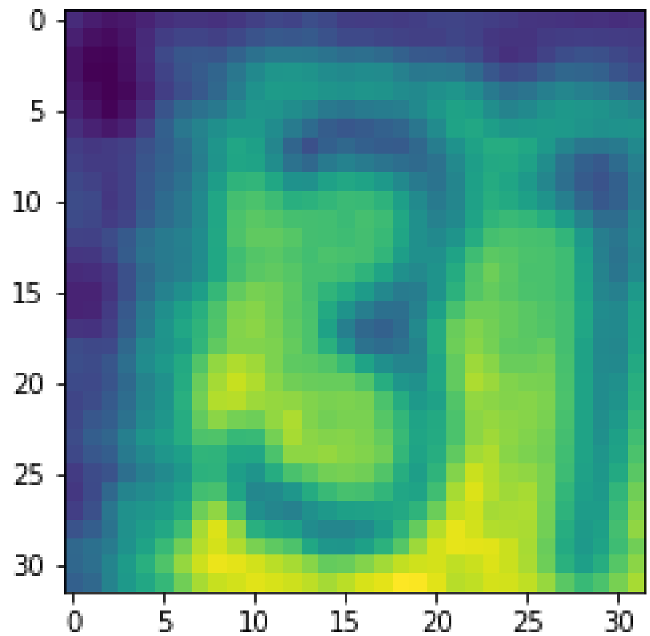
[5]



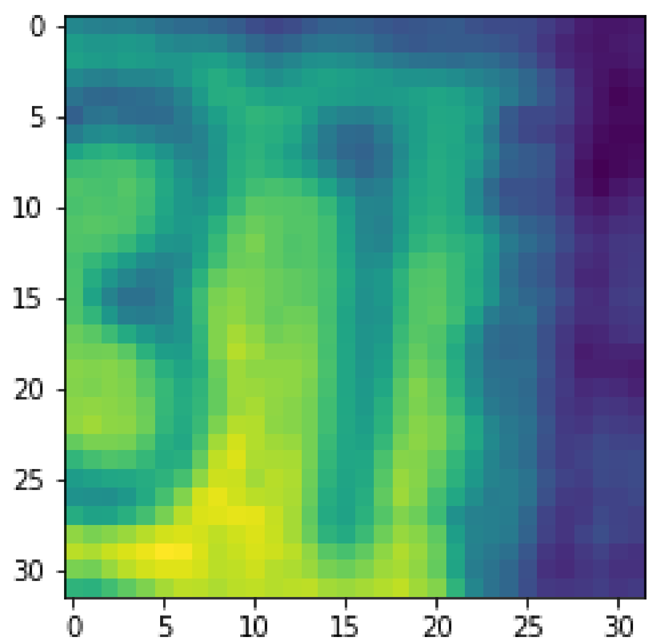
[9]



[3]



[3]



[1]

```
In [13]: X_train[0].shape
```

```
Out[13]: (32, 32, 3)
```

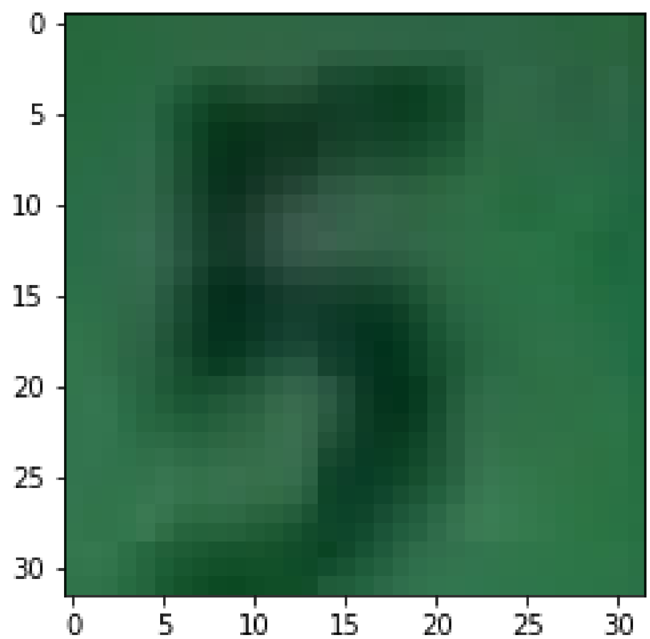
```
In [14]: from sklearn.preprocessing import OneHotEncoder

         enc = OneHotEncoder().fit(y_train)
         y_train_oh = enc.transform(y_train).toarray()
         y_test_oh = enc.transform(y_test).toarray()
```

```
In [15]: y_test_oh[0]
```

```
Out[15]: array([0., 0., 0., 0., 1., 0., 0., 0., 0., 0.])
```

```
In [16]: plt.imshow(X_test[0])
```

```
Out[16]: <matplotlib.image.AxesImage at 0x15e267351c8>
```



1.3 2. MLP neural network classifier

- Build an MLP classifier model using the Sequential API. Your model should use only Flatten and Dense layers, with the final layer having a 10-way softmax output.
- You should design and build the model yourself. Feel free to experiment with different MLP architectures. *Hint: to achieve a reasonable accuracy you won't need to use more than 4 or 5 layers.*
- Print out the model summary (using the summary() method)
- Compile and train the model (we recommend a maximum of 30 epochs), making use of both training and validation sets during the training run.

- Your model should track at least one appropriate metric, and use at least two callbacks during training, one of which should be a ModelCheckpoint callback.
- As a guide, you should aim to achieve a final categorical cross entropy training loss of less than 1.0 (the validation loss might be higher).
- Plot the learning curves for loss vs epoch and accuracy vs epoch for both training and validation sets.
- Compute and display the loss and accuracy of the trained model on the test set.

In [17]:
```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
```

In [42]:
```
checkpoint = ModelCheckpoint(filepath = 'SeqMode\\mySeqModel', save_best_only=True, sa
earlystop = EarlyStopping(patience=5, monitor='loss')
```

In [ ]:

In [43]:
```
model2 = Sequential([
        Flatten(input_shape=X_train[0].shape),
        Dense(128*4, activation='relu'),
        Dense(64, activation='relu'),
        BatchNormalization(),
        Dense(64, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        Dense(32, activation='relu'),
        Dense(10, activation='softmax')
])
model2.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_4 (Flatten)          (None, 3072)              0
_________________________________________________________________
dense_18 (Dense)             (None, 512)               1573376
_________________________________________________________________
dense_19 (Dense)             (None, 64)                32832
_________________________________________________________________
batch_normalization_4 (Batch (None, 64)                256
_________________________________________________________________
dense_20 (Dense)             (None, 64)                4160
_________________________________________________________________
dropout_5 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_21 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_22 (Dense)             (None, 10)                330
=================================================================
Total params: 1,613,034
Trainable params: 1,612,906
```

```
Non-trainable params: 128
_________________________________________________________________
```

```
In [44]: model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])

In [45]: history = model2.fit(X_train, y_train_oh, callbacks=[checkpoint, earlystop], batch_siz
```

```
Epoch 1/30
WARNING:tensorflow:Unresolved object in checkpoint: (root).layer-10
WARNING:tensorflow:Unresolved object in checkpoint: (root).layer_with_weights-6
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer
WARNING:tensorflow:Unresolved object in checkpoint: (root).layer_with_weights-6.kernel
WARNING:tensorflow:Unresolved object in checkpoint: (root).layer_with_weights-6.bias
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or tf.keras.Mode
```

```
WARNING:tensorflow:Unresolved object in checkpoint: (root).layer-10
WARNING:tensorflow:Unresolved object in checkpoint: (root).layer_with_weights-6
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer
WARNING:tensorflow:Unresolved object in checkpoint: (root).layer_with_weights-6.kernel
WARNING:tensorflow:Unresolved object in checkpoint: (root).layer_with_weights-6.bias
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.learning_rate
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'm' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer's state 'v' for (root).lay
WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore or tf.keras.Mode
571/573 [============================>.] - ETA: 0s - loss: 1.9465 - acc: 0.3090
Epoch 00001: val_loss improved from inf to 4.97246, saving model to SeqMode\mySeqModel
573/573 [==============================] - 5s 9ms/step - loss: 1.9464 - acc: 0.3093 - val_loss
Epoch 2/30
569/573 [============================>.] - ETA: 0s - loss: 1.5056 - acc: 0.5030
Epoch 00002: val_loss improved from 4.97246 to 2.01789, saving model to SeqMode\mySeqModel
573/573 [==============================] - 10s 18ms/step - loss: 1.5053 - acc: 0.5031 - val_los
Epoch 3/30
572/573 [============================>.] - ETA: 0s - loss: 1.4056 - acc: 0.5435
```

```
Epoch 00003: val_loss improved from 2.01789 to 1.55750, saving model to SeqMode\mySeqModel
573/573 [==============================] - 7s 13ms/step - loss: 1.4055 - acc: 0.5436 - val_loss
Epoch 4/30
572/573 [============================>.] - ETA: 0s - loss: 1.3618 - acc: 0.5625
Epoch 00004: val_loss improved from 1.55750 to 1.42611, saving model to SeqMode\mySeqModel
573/573 [==============================] - 7s 13ms/step - loss: 1.3620 - acc: 0.5625 - val_loss
Epoch 5/30
570/573 [============================>.] - ETA: 0s - loss: 1.3240 - acc: 0.5783
Epoch 00005: val_loss did not improve from 1.42611
573/573 [==============================] - 7s 12ms/step - loss: 1.3240 - acc: 0.5784 - val_loss
Epoch 6/30
568/573 [============================>.] - ETA: 0s - loss: 1.3009 - acc: 0.5864
Epoch 00006: val_loss did not improve from 1.42611
573/573 [==============================] - 7s 12ms/step - loss: 1.3007 - acc: 0.5866 - val_loss
Epoch 7/30
569/573 [============================>.] - ETA: 0s - loss: 1.2678 - acc: 0.5993
Epoch 00007: val_loss improved from 1.42611 to 1.42144, saving model to SeqMode\mySeqModel
573/573 [==============================] - 7s 12ms/step - loss: 1.2672 - acc: 0.5993 - val_loss
Epoch 8/30
570/573 [============================>.] - ETA: 0s - loss: 1.2499 - acc: 0.6053
Epoch 00008: val_loss did not improve from 1.42144
573/573 [==============================] - 7s 12ms/step - loss: 1.2496 - acc: 0.6055 - val_loss
Epoch 9/30
572/573 [============================>.] - ETA: 0s - loss: 1.2275 - acc: 0.6134
Epoch 00009: val_loss did not improve from 1.42144
573/573 [==============================] - 7s 12ms/step - loss: 1.2276 - acc: 0.6134 - val_loss
Epoch 10/30
573/573 [==============================] - ETA: 0s - loss: 1.2067 - acc: 0.6202
Epoch 00010: val_loss did not improve from 1.42144
573/573 [==============================] - 7s 12ms/step - loss: 1.2067 - acc: 0.6202 - val_loss
Epoch 11/30
572/573 [============================>.] - ETA: 0s - loss: 1.1956 - acc: 0.6249
Epoch 00011: val_loss improved from 1.42144 to 1.27431, saving model to SeqMode\mySeqModel
573/573 [==============================] - 7s 13ms/step - loss: 1.1957 - acc: 0.6249 - val_loss
Epoch 12/30
573/573 [==============================] - ETA: 0s - loss: 1.1764 - acc: 0.6298
Epoch 00012: val_loss did not improve from 1.27431
573/573 [==============================] - 7s 12ms/step - loss: 1.1764 - acc: 0.6298 - val_loss
Epoch 13/30
570/573 [============================>.] - ETA: 0s - loss: 1.1636 - acc: 0.6356
Epoch 00013: val_loss did not improve from 1.27431
573/573 [==============================] - 7s 12ms/step - loss: 1.1632 - acc: 0.6357 - val_loss
Epoch 14/30
570/573 [============================>.] - ETA: 0s - loss: 1.1516 - acc: 0.6387
Epoch 00014: val_loss did not improve from 1.27431
573/573 [==============================] - 7s 12ms/step - loss: 1.1514 - acc: 0.6388 - val_loss
Epoch 15/30
571/573 [============================>.] - ETA: 0s - loss: 1.1799 - acc: 0.6270
```

```
Epoch 00015: val_loss improved from 1.27431 to 1.27116, saving model to SeqMode\mySeqModel
573/573 [==============================] - 7s 12ms/step - loss: 1.1800 - acc: 0.6270 - val_loss
Epoch 16/30
572/573 [============================>.] - ETA: 0s - loss: 1.1662 - acc: 0.6328
Epoch 00016: val_loss improved from 1.27116 to 1.25893, saving model to SeqMode\mySeqModel
573/573 [==============================] - 7s 12ms/step - loss: 1.1661 - acc: 0.6327 - val_loss
Epoch 17/30
571/573 [============================>.] - ETA: 0s - loss: 1.1407 - acc: 0.6432
Epoch 00017: val_loss did not improve from 1.25893
573/573 [==============================] - 7s 12ms/step - loss: 1.1408 - acc: 0.6431 - val_loss
Epoch 18/30
569/573 [============================>.] - ETA: 0s - loss: 1.1194 - acc: 0.6506
Epoch 00018: val_loss did not improve from 1.25893
573/573 [==============================] - 7s 12ms/step - loss: 1.1190 - acc: 0.6506 - val_loss
Epoch 19/30
568/573 [============================>.] - ETA: 0s - loss: 1.1128 - acc: 0.6503
Epoch 00019: val_loss improved from 1.25893 to 1.16105, saving model to SeqMode\mySeqModel
573/573 [==============================] - 7s 12ms/step - loss: 1.1127 - acc: 0.6505 - val_loss
Epoch 20/30
571/573 [============================>.] - ETA: 0s - loss: 1.1038 - acc: 0.6555
Epoch 00020: val_loss improved from 1.16105 to 1.14658, saving model to SeqMode\mySeqModel
573/573 [==============================] - 7s 12ms/step - loss: 1.1035 - acc: 0.6555 - val_loss
Epoch 21/30
571/573 [============================>.] - ETA: 0s - loss: 1.0951 - acc: 0.6569
Epoch 00021: val_loss did not improve from 1.14658
573/573 [==============================] - 7s 12ms/step - loss: 1.0949 - acc: 0.6571 - val_loss
Epoch 22/30
570/573 [============================>.] - ETA: 0s - loss: 1.0894 - acc: 0.6605
Epoch 00022: val_loss did not improve from 1.14658
573/573 [==============================] - 7s 12ms/step - loss: 1.0896 - acc: 0.6603 - val_loss
Epoch 23/30
568/573 [============================>.] - ETA: 0s - loss: 1.0804 - acc: 0.6629
Epoch 00023: val_loss improved from 1.14658 to 1.06969, saving model to SeqMode\mySeqModel
573/573 [==============================] - 7s 13ms/step - loss: 1.0811 - acc: 0.6627 - val_loss
Epoch 24/30
571/573 [============================>.] - ETA: 0s - loss: 1.0805 - acc: 0.6635
Epoch 00024: val_loss did not improve from 1.06969
573/573 [==============================] - 7s 12ms/step - loss: 1.0809 - acc: 0.6633 - val_loss
Epoch 25/30
568/573 [============================>.] - ETA: 0s - loss: 1.0692 - acc: 0.6673
Epoch 00025: val_loss did not improve from 1.06969
573/573 [==============================] - 7s 12ms/step - loss: 1.0690 - acc: 0.6674 - val_loss
Epoch 26/30
570/573 [============================>.] - ETA: 0s - loss: 1.0605 - acc: 0.6690
Epoch 00026: val_loss did not improve from 1.06969
573/573 [==============================] - 7s 12ms/step - loss: 1.0605 - acc: 0.6691 - val_loss
Epoch 27/30
568/573 [============================>.] - ETA: 0s - loss: 1.0483 - acc: 0.6751
```

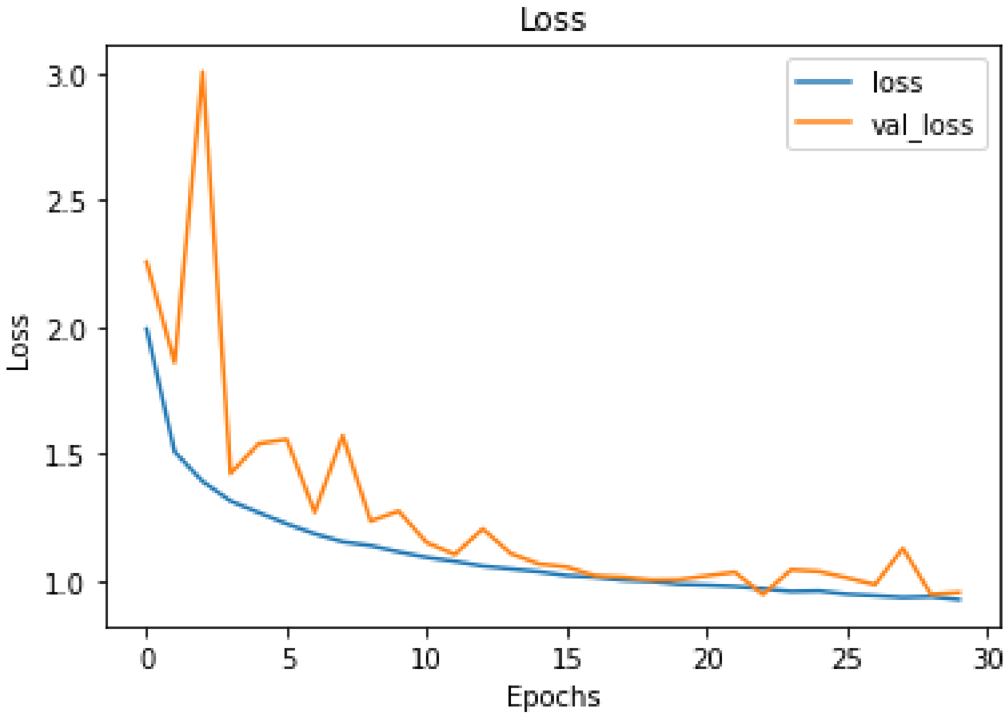
```
Epoch 00027: val_loss improved from 1.06969 to 1.05753, saving model to SeqMode\mySeqModel
573/573 [==============================] - 7s 12ms/step - loss: 1.0482 - acc: 0.6752 - val_loss
Epoch 28/30
570/573 [============================>.] - ETA: 0s - loss: 1.0453 - acc: 0.6760
Epoch 00028: val_loss did not improve from 1.05753
573/573 [==============================] - 7s 12ms/step - loss: 1.0456 - acc: 0.6759 - val_loss
Epoch 29/30
569/573 [============================>.] - ETA: 0s - loss: 1.0449 - acc: 0.6768
Epoch 00029: val_loss did not improve from 1.05753
573/573 [==============================] - 7s 12ms/step - loss: 1.0448 - acc: 0.6768 - val_loss
Epoch 30/30
569/573 [============================>.] - ETA: 0s - loss: 1.0398 - acc: 0.6764
Epoch 00030: val_loss did not improve from 1.05753
573/573 [==============================] - 7s 12ms/step - loss: 1.0399 - acc: 0.6763 - val_loss
```

In [22]:
```
!dir
```

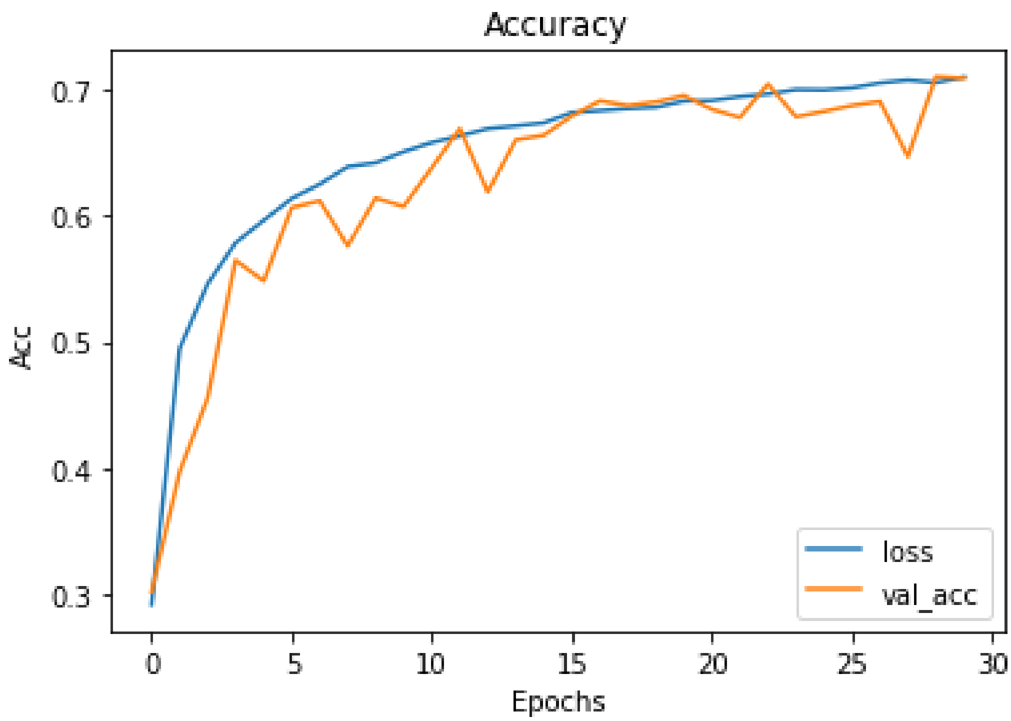
```
 Volume in drive C has no label.
 Volume Serial Number is 8821-EC45

 Directory of C:\Users\Ahmad Mustafa Anis\Desktop\Getting started with TF 2\Capstone Project

07/07/2020  10:42 AM    <DIR>          .
07/07/2020  10:42 AM    <DIR>          ..
07/07/2020  10:40 AM    <DIR>          .ipynb_checkpoints
07/07/2020  10:42 AM           254,721 Capstone Project.ipynb
07/07/2020  10:42 AM                77 checkpoint
07/07/2020  10:39 AM    <DIR>          data
07/07/2020  10:42 AM             5,540 mySeqModel.data-00000-of-00002
07/07/2020  10:42 AM        19,355,408 mySeqModel.data-00001-of-00002
07/07/2020  10:42 AM             3,037 mySeqModel.index
               5 File(s)     19,618,783 bytes
               4 Dir(s)  80,843,591,680 bytes free
```

In [24]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['loss','val_loss'], loc='upper right')
plt.title("Loss")
```

Out[24]:
```
Text(0.5, 1.0, 'Loss')
```




```
In [26]: plt.plot(history.history['acc'])
         plt.plot(history.history['val_acc'])
         plt.xlabel('Epochs')
         plt.ylabel('Acc')
         plt.legend(['loss','val_acc'], loc='lower right')
         plt.title("Accuracy")

Out[26]: Text(0.5, 1.0, 'Accuracy')
```



1.4 3. CNN neural network classifier

- Build a CNN classifier model using the Sequential API. Your model should use the Conv2D, MaxPool2D, BatchNormalization, Flatten, Dense and Dropout layers. The final layer should again have a 10-way softmax output.
- You should design and build the model yourself. Feel free to experiment with different CNN architectures. *Hint: to achieve a reasonable accuracy you won't need to use more than 2 or 3 convolutional layers and 2 fully connected layers.)*
- The CNN model should use fewer trainable parameters than your MLP model.
- Compile and train the model (we recommend a maximum of 30 epochs), making use of both training and validation sets during the training run.
- Your model should track at least one appropriate metric, and use at least two callbacks during training, one of which should be a ModelCheckpoint callback.
- You should aim to beat the MLP model performance with fewer parameters!
- Plot the learning curves for loss vs epoch and accuracy vs epoch for both training and validation sets.
- Compute and display the loss and accuracy of the trained model on the test set.

```
In [27]: model3 = Sequential([
             Conv2D(filters= 16, kernel_size= 3, activation='relu', input_shape=X_train[0].sha
             MaxPool2D(pool_size= (3,3), strides=1),
             Conv2D(filters= 32, kernel_size = 3, padding='valid', strides=1, activation='relu
             MaxPool2D(pool_size = (1,1), strides = 3),
             BatchNormalization(),
```

```
            Conv2D(filters= 32, kernel_size = 3, padding='valid', strides=2, activation='relu
            tf.keras.layers.Dropout(0.5),
            Flatten(),
            Dense(128, activation='relu'),
            Dense(32, activation='relu'),
            tf.keras.layers.Dropout(0.3),
            Dense(10, activation='softmax')
        ])
```

```
In [28]: model3.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 16)        448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 28, 28, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 26, 26, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 9, 9, 32)          0
_________________________________________________________________
batch_normalization_1 (Batch (None, 9, 9, 32)          128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 32)          9248
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_6 (Dense)              (None, 32)                4128
_________________________________________________________________
dropout_2 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                330
=================================================================
Total params: 84,586
Trainable params: 84,522
Non-trainable params: 64
_________________________________________________________________
```

```
In [29]: ## Less parameters then normal model
```

```
In [30]: model3.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])
```

```
In [31]: callback1 = ModelCheckpoint(filepath='CNNweights', save_best_only=True, save_weights_
         callback2 = EarlyStopping(monitor='loss',patience=7, verbose=1)
```

```
In [32]: X_train.shape
```

```
Out[32]: (73257, 32, 32, 3)
```

```
In [33]: history = model3.fit(X_train, y_train_oh, callbacks=[checkpoint, earlystop], batch_si
```

```
Epoch 1/30
287/287 [==============================] - 24s 82ms/step - loss: 1.7868 - acc: 0.3753 - val_lo
Epoch 2/30
287/287 [==============================] - 23s 79ms/step - loss: 0.9361 - acc: 0.7078 - val_lo
Epoch 3/30
287/287 [==============================] - 23s 81ms/step - loss: 0.7655 - acc: 0.7679 - val_lo
Epoch 4/30
287/287 [==============================] - 23s 80ms/step - loss: 0.6836 - acc: 0.7937 - val_lo
Epoch 5/30
287/287 [==============================] - 23s 79ms/step - loss: 0.6442 - acc: 0.8064 - val_lo
Epoch 6/30
287/287 [==============================] - 23s 79ms/step - loss: 0.5972 - acc: 0.8218 - val_lo
Epoch 7/30
287/287 [==============================] - 23s 78ms/step - loss: 0.5705 - acc: 0.8314 - val_lo
Epoch 8/30
287/287 [==============================] - 22s 78ms/step - loss: 0.5490 - acc: 0.8380 - val_lo
Epoch 9/30
287/287 [==============================] - 22s 78ms/step - loss: 0.5277 - acc: 0.8431 - val_lo
Epoch 10/30
287/287 [==============================] - 23s 79ms/step - loss: 0.5094 - acc: 0.8495 - val_lo
Epoch 11/30
287/287 [==============================] - 23s 79ms/step - loss: 0.4962 - acc: 0.8530 - val_lo
Epoch 12/30
287/287 [==============================] - 22s 78ms/step - loss: 0.4857 - acc: 0.8568 - val_lo
Epoch 13/30
287/287 [==============================] - 22s 78ms/step - loss: 0.4724 - acc: 0.8615 - val_lo
Epoch 14/30
287/287 [==============================] - 23s 79ms/step - loss: 0.4646 - acc: 0.8621 - val_lo
Epoch 15/30
287/287 [==============================] - 22s 78ms/step - loss: 0.4572 - acc: 0.8667 - val_lo
Epoch 16/30
287/287 [==============================] - 22s 78ms/step - loss: 0.4496 - acc: 0.8665 - val_lo
Epoch 17/30
287/287 [==============================] - 23s 79ms/step - loss: 0.4485 - acc: 0.8686 - val_lo
Epoch 18/30
287/287 [==============================] - 22s 78ms/step - loss: 0.4357 - acc: 0.8731 - val_lo
Epoch 19/30
287/287 [==============================] - 22s 78ms/step - loss: 0.4384 - acc: 0.8721 - val_lo
Epoch 20/30
287/287 [==============================] - 22s 78ms/step - loss: 0.4298 - acc: 0.8724 - val_lo
```

```
Epoch 21/30
287/287 [==============================] - 22s 78ms/step - loss: 0.4163 - acc: 0.8776 - val_lo
Epoch 22/30
287/287 [==============================] - 22s 78ms/step - loss: 0.4094 - acc: 0.8794 - val_lo
Epoch 23/30
287/287 [==============================] - 22s 78ms/step - loss: 0.4077 - acc: 0.8806 - val_lo
Epoch 24/30
287/287 [==============================] - 23s 78ms/step - loss: 0.4064 - acc: 0.8811 - val_lo
Epoch 25/30
287/287 [==============================] - 23s 79ms/step - loss: 0.4024 - acc: 0.8828 - val_lo
Epoch 26/30
287/287 [==============================] - 23s 79ms/step - loss: 0.4032 - acc: 0.8824 - val_lo
Epoch 27/30
287/287 [==============================] - 23s 80ms/step - loss: 0.3952 - acc: 0.8847 - val_lo
Epoch 28/30
287/287 [==============================] - 23s 79ms/step - loss: 0.3913 - acc: 0.8843 - val_lo
Epoch 29/30
287/287 [==============================] - 23s 78ms/step - loss: 0.3879 - acc: 0.8861 - val_lo
Epoch 30/30
287/287 [==============================] - 23s 78ms/step - loss: 0.3871 - acc: 0.8857 - val_lo
```

1.4.1 We can see that we improved our accuracy very much ascompared normal dense model in 4 epochs while having very less parameters

1.5 4. Get model predictions

- Load the best weights for the MLP and CNN models that you saved during the training run.
- Randomly select 5 images and corresponding labels from the test set and display the images with their labels.
- Alongside the image and label, show each model's predictive distribution as a bar chart, and the final model prediction given by the label with maximum probability.

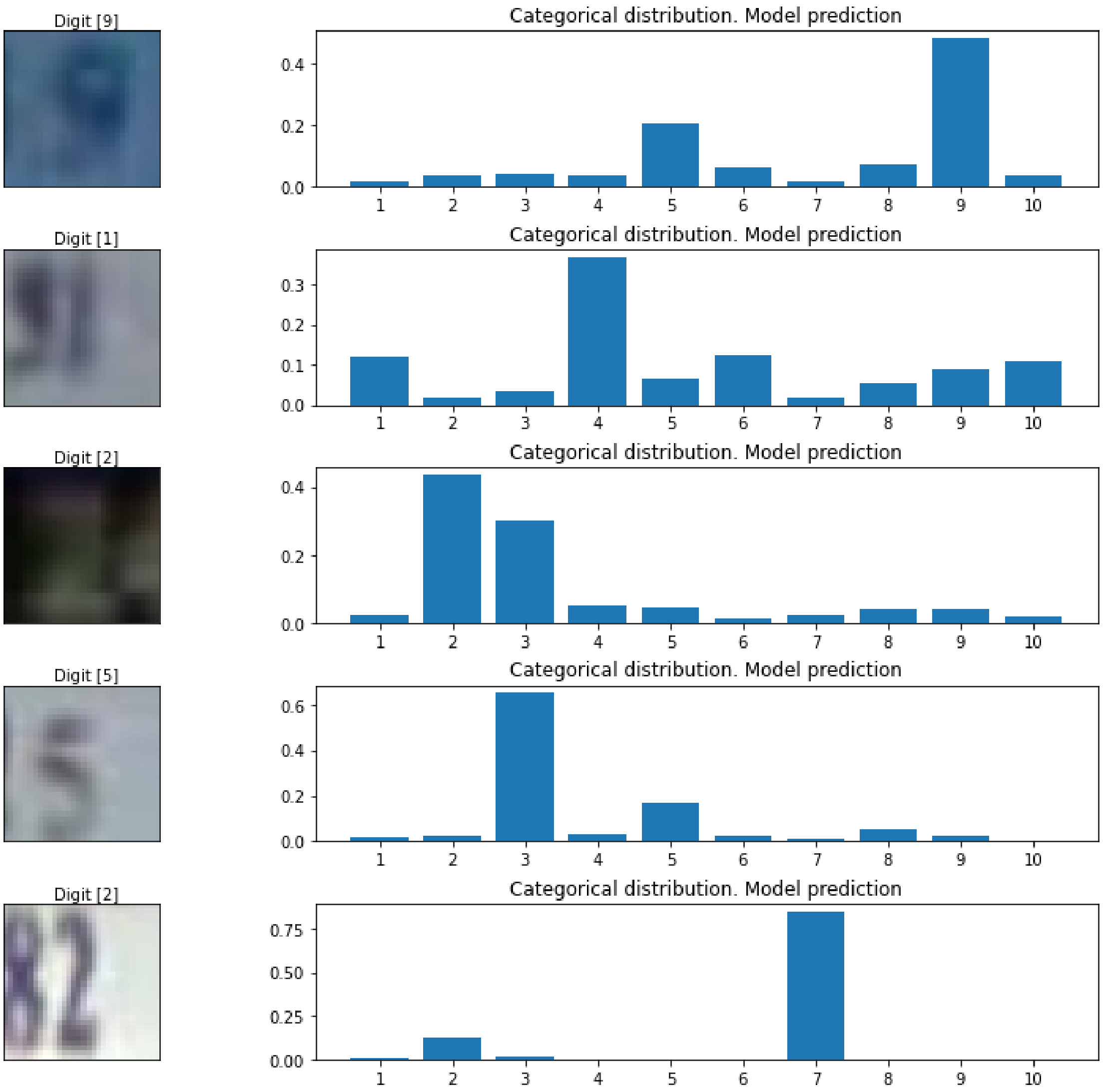
In [46]:
```python
model2.load_weights('SeqMode\\mySeqModel')
```

Out[46]:
```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x15e7229dd48>
```

In [49]:
```python
import random
```

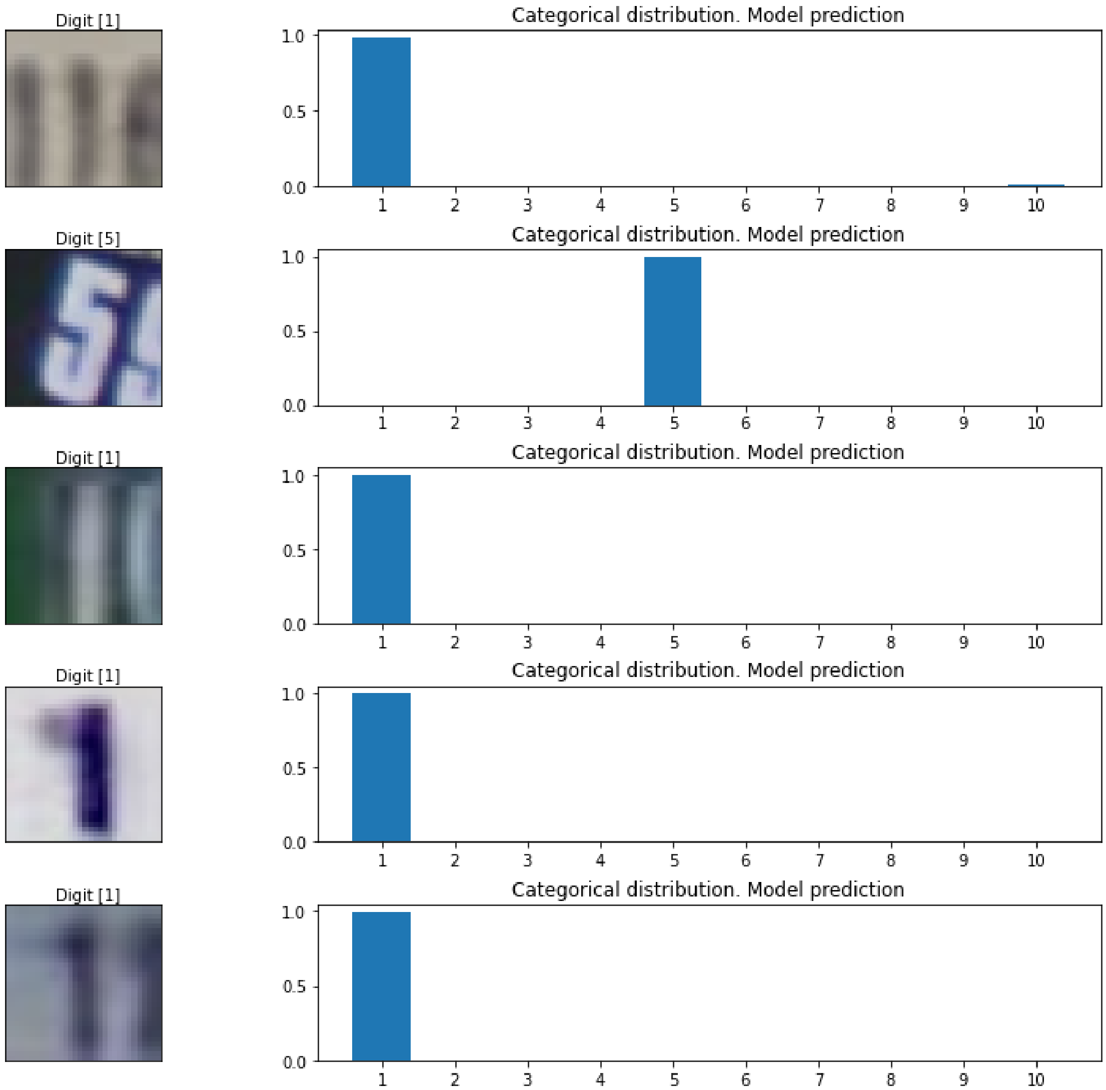
In [67]:
```python
num_test_images = X_test.shape[0]

random_inx = np.random.choice(num_test_images, 5)
random_test_images = X_test[random_inx, ...]
random_test_labels = y_test[random_inx, ...]

predictions = model2.predict(random_test_images)

fig, axes = plt.subplots(5, 2, figsize=(16, 12))
fig.subplots_adjust(hspace=0.4, wspace=-0.2)
```

```
for i, (prediction, image, label) in enumerate(zip(predictions, random_test_images, ra
    axes[i, 0].imshow(np.squeeze(image))
    axes[i, 0].get_xaxis().set_visible(False)
    axes[i, 0].get_yaxis().set_visible(False)
    axes[i, 0].text(10., -1.5, f'Digit {label}')
    axes[i, 1].bar(np.arange(1,11), prediction)
    axes[i, 1].set_xticks(np.arange(1,11))
    axes[i, 1].set_title("Categorical distribution. Model prediction")

plt.show()
```




```
In [69]: num_test_images = X_test.shape[0]
```

```
random_inx = np.random.choice(num_test_images, 5)
random_test_images = X_test[random_inx, ...]
random_test_labels = y_test[random_inx, ...]

predictions = model3.predict(random_test_images)

fig, axes = plt.subplots(5, 2, figsize=(16, 12))
fig.subplots_adjust(hspace=0.4, wspace=-0.2)

for i, (prediction, image, label) in enumerate(zip(predictions, random_test_images, ra
    axes[i, 0].imshow(np.squeeze(image))
    axes[i, 0].get_xaxis().set_visible(False)
    axes[i, 0].get_yaxis().set_visible(False)
    axes[i, 0].text(10., -1.5, f'Digit {label}')
    axes[i, 1].bar(np.arange(1,11), prediction)
    axes[i, 1].set_xticks(np.arange(1,11))
    axes[i, 1].set_title("Categorical distribution. Model prediction")

plt.show()
```


Digit [1]
Categorical distribution. Model prediction
Digit [5]
Categorical distribution. Model prediction
Digit [1]
Categorical distribution. Model prediction
Digit [1]
Categorical distribution. Model prediction
Digit [1]
Categorical distribution. Model prediction
1.0
0.5
0.0
1 2 3 4 5 6 7 8 9 10

In [ ]:

In [ ]: